IQLR

International QL Report

ISSN 1078-5787

Volume 4 Issue 6
March/April
1995

Youngest Known QL Programmer

Hard At Work Developing What Is Said To Be A Rival Product To Minerva

Digital Precision's

'Michelle Vachha' With Her Trusty Super Gold Card Expanded QL

IQLR.....

International QL Report is published by:

IQLR
P. O. Box 3991
Newport, RI 02840-0987
USA

IQLR
23 Ben Culey Drive
Thetford, Norfolk IP24 1QJ
GREAT BRITAIN

Tel/Fax: +1 401 849 3805

PUBLISHER: Robert Dyl, Sr. (ISSN 1078-5787)

IQLR is published bi-monthly, our volume year begins on 1 May and runs through 30 April. Subscriptions begin with the current issue at the time of sign up. Subscription rates are as follows:

USA	$24.00 per year
British Isles & Europe	£25.00 per year
Canada	$27.00 (US Funds)
Central/South America	$34.00 (US Funds)
Rest of World	$40.00 (US Funds)

UK and European readers may send their subscriptions to our European office listed above. Postal, Euro, Bank and Personal Cheques in Pounds Sterling, drawn on a UK bank should be made payable to IQLR.

Payment in US $ can be made by either a Postal, Bank or Personal cheque (drawn on a US BANK) or bank notes (£ or DM equivalent to the US $ amount) should be sent to our North American office.

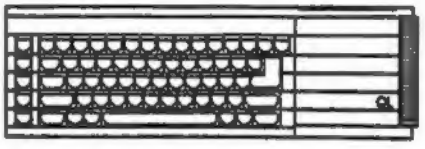
We welcome your comments, suggestions and articles. YOU make IQLR possible. We are constantly changing and adjusting to meet your needs and requirements. Articles submitted for publication should be on a 3.5" disk in Quill or Text87 format. To enhance your article you may wish to include Saved Screen dumps. PLEASE send a hard copy of all screens to be included, don't forget to specify where in the text you would like the screens placed.

Article and Advertising DEADLINES are as follows:

Issue 1	10 April
Issue 2	10 June
Issue 3	10 August
Issue 4	10 October
Issue 5	10 December
Issue 6	10 February

IQLR reserves the right to publish or not publish any material submitted. Under no circumstances will IQLR be held liable for any direct, indirect or consequential damage or loss arising out of the use and/or inability to use any of the material published in IQLR. The opinions expressed herein are those of the authors and are not necessarily those of the publisher.

Contents.....



Advertisers.....

Notes from the Publisher

Newport, Rhode Island, USA - Bob Dyl

It's with great anticipation that we end our fourth year of publication and begin work on our fifth. The number of International QL shows continues to grow, more and more high quality PD software is finding its way into users hands, more hardware developments than we've seen in years are about to break on the scene and new exciting software is not far behind. Couple all this with a new COMPATIBLE operating system that will take us into the next century and what you have is a vibrant community back on the cutting edge.

IQLR ENDORSMENT - In the past, we have avoided endorsing any particular product, but at this time we believe that one has come along that is so spectacular and important to the continuance, promotion and expansion of our community that an endorsement is more than required.

Many of you have invested your hard earned money on products such as: the Gold Card, Super Gold Card, QXL, QBIDE hard disk interface and are awaiting some or all of the developments Nasta wrote about in our last issue. Then why not cap it off by purchasing the one product that is the natural while compatible next step into the future? We have, and we are totally satisfied that it is by far the greatest development since QDOS itself.

IQLR endorses Tony Tebby's SMSQ/E. Why ? Because unlike other operating systems, it has not become obsolete on its launch, but will continue to grow and develope, just as you and I continue on with the system we love.

DILWYN JONES - It's with mixed emotions that we inform you of Dilwyn Jones decision to stop trading. Dilwyn's decision is based on personal reasons (in which as a father and grandfather I can concur) rather than economical ones. As a matter of fact he states, that DJC has been more profitable than ever. So, as of the end of March 1995, Dilwyn Jones Computing will fade into the past. But, NOT DILWYN, he intends to continue writing software and articles for the QL press. Dilwyn is offering great deals on DJC software (while current stocks last), so if you haven't received a mailing from him, contact him quickly (see DJC adverts for address and telephone/fax number).

MINI-MART of VALUES - IQLR has introduced a new program (see advert in this issue) that will allow Independent Producers (be it producers of one program or many) an economical way to present their product(s) to the QDOS/SMSQ community . For information contact me at our North American office.

General IQLR Information - if you received an IQLR leaflet with this issue, it is to inform you that your subscription has now lapsed. Why not take the time NOW and fill it out and post it back to us, it'll guarantee that you won't experience a delay on receiving your next issue.

Back issues always seems to be a problem; our policy is to print about 100 copies over what we require, when they go the issue is out of print. We have tried to get someone interested in supplying back issues but when they realize the work involved, they quickly decline. You may notice, three personal logo's or clipart in this issue, starting with my own. If you have a personal logo or piece of clipart that is identified with you, send it in with your articles and we'll include them along with your article.

ERGON DEVELOPMENT - have recently changed their (voice) telephone number to: +39 522 300409 please note their adverts elsewhere in this issue.

DIGITAL PRECISION - has delivered on its promise to award the first IQLR subscriber who ordered their software deal with £100 (value) worth of hardware. The lucky person was H W Frogatt of Tunbridgewell, UK. *(How about letting us know what Freddy sent you?)*

Notes from the Publisher - (CONT'D)

3rd Annual North American QL Show will be held on Saturday the 10th of June 1995 in OAK RIDGE, TENNESSEE, USA. Those who have attended in the past, will tell you that the show is really secondary to the real event, the GREAT OLD-FASHIONED GOOD TIME we all have. Many of us arrive two days in advance and stay one or two days afterwards, others stay a day or two while still others arrive the day of the show. No matter how long your with us, you will have a great time.

Traders expected to attend include: Stuart Honeyball of Miracle Systems (UK), Tony Firshman of TF Services (UK), Bill Richardson of W N Richardson and Co (UK), Jochen Merz of Jochen Merz Software (GERMANY), Frank Davis of Mechanical Affinity (USA), Carol and Frank Davis of Update Magazine (USA), Bill Cable of Wood and Wind Computing (USA) and while not traders, John Impellizzeri and Don Walterman of QBOX-USA (USA) will be demonstrating their QL Bulletin Board.

NEW PRODUCTS and OLD are expected to abound, including: the Masterpiece Enhanced Graphics Card - Super Gold Card - QXLs from Miracle Systems, Super Hermes - Minerva - I2C interfaces from TF Services, and Mechanical Affinity will have Qubide hard disk interfaces and just about anything else you might want.

Jochen Merz will be demonstrating SMSQ/E plus a number of his PE compatible programs including some new products. Wood and Wind Computing will be demonstrating their state of the art financial package QLerk along with their other products and Mechanical Affinity if they are true to form, will have just about every program currently available for the QL plus plenty of spare parts.

For the first time we'll have a table set up for Trade and Sale, so bring those excess items you have with you, people are always looking for good buys.

As has been our practice, we will have a Dutch Treat Dinner (you pay for your own meal) following the show. The Registration Fee will be $3 (US) per person in advance and $5 (US) at the door. To register for the show (not the motel) contact IQLR at our North American office.

BASE OF OPERATIONS will be the ***SUPER 8 MOTEL*** 1590 Oak Ridge Turnpike Oak Ridge, TN, USA. The room rates are $37 for a single and $41 for a double plus local taxes. Room rental includes the use of their outdoor swimming pool, 25" color television (in each room) with multi-channel Cable, and a FREE Continental Breakfast. Please make your reservations well in advance of the show and mention that your part of the IQLR group, as they are holding a limited (20) number of rooms at the rates listed above. FOR RESERVATIONS TELEPHONE: US 615 483 1200

The venue of the show (with plenty of free parking) will be just down the road from the motel at : FAITH LUTHERN CHURCH, 1300 Oak Ridge Turnpike, Oak Ridge,TN . Show hours will be: 10 am (EST) to 4pm (EST).

INTERNATIONAL TRAVELLERS should book their flights into NASHVILLE ,TN if at all possible, there is then a three hour Coach ride to Oak Ridge. An alternative route would be to land in Atlanta, Georgia then take a 5 hour Coach ride to Oak Ridge.

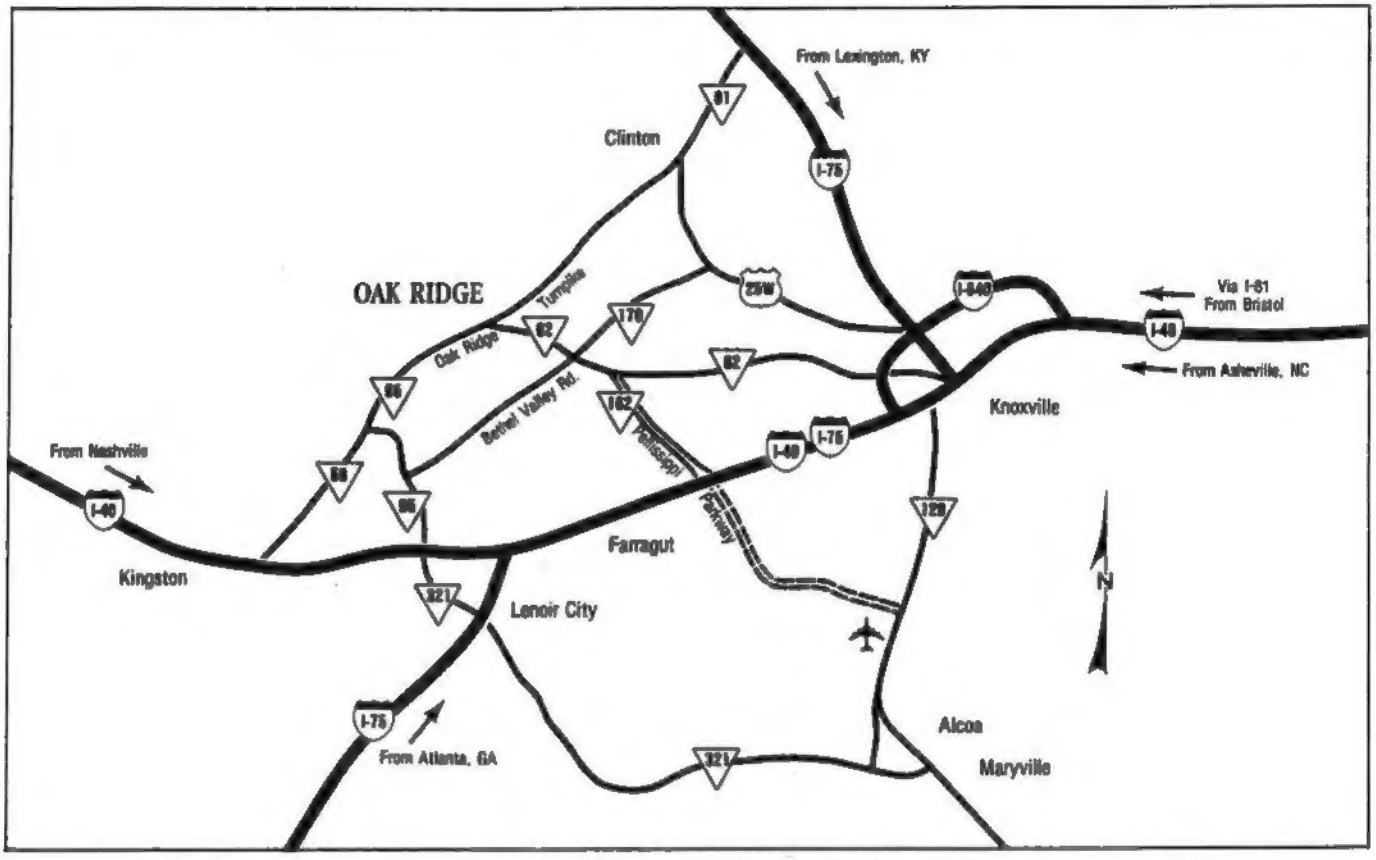
Internal flights should land in Knoxville, TN then there is a short 20 minute ride to Oak Ridge.

For those of you driving to Oak Ridge we have included (on the next page) a Location Map with a Profile of Oak Ridge and a list of Transportation Services compiled and provided by The Oak Ridge Convention and Visitors Bureau.

For additional information concerning the area and transportation connections please contact:

Mel LaVerne
103 Endicott Lane
Oak Ridge, Tennessee 37830-4117 USA
Telephone: +1 615 483 4153

Location Map



The above map highlights routes from the interstates and the Knoxville regional airport. Oak Ridge is located ten miles west of Knoxville, less than 45 minutes from McGhee-Tyson Airport and 15 minutes from Interstates 40 and 75. Two-thirds of the U. S. population is within one day's drive of Oak Ridge.

Profile of Oak Ridge

Population 30,000
Area ..... 92 square miles
Climate ..... Moderate, four seasons
Average temperature ..... Winter 39° F, Summer 75° F
Average precipitation ..... 55 inches
Average humidity 70%

Attractions:
- American Museum of Science and Energy
- Historic Graphite Reactor
- The Arboretum
- Children's Museum of Oak Ridge
- Museum of Fine Arts

Recreation:
- Hiking trails and jogging tracks
- Two golf courses
- Indoor and outdoor public swimming pools
- Lighted softball fields and tennis courts
- City parks and recreation areas
- A 2,000-meter rowing course on Melton Lake

Transportation Services

McGhee-Tyson Airport - Airlines servicing the area include: American Eagle, ComAir, Delta, Northwest, Transworld, United and USAir.

Limousine Service
Oak Ridge Ballroom Limousine
66½ Outer Drive
Oak Ridge, TN 37830
(615) 483-3800

Knoxville Limousine Service
McGhee-Tyson Airport
(615) 970-7007
(615) 970-3349
(offers shuttle service also)

Charter Bus Services
Wilson Tours
P.O. Box 452
Hwy 61 at Indian Hills Dr.
Clinton, TN 37717
(615) 457-1643

Clinton Bus Co.
930 Charles Seivers Blvd.
Clinton, TN 37716
(615) 457-7538
(615) 689-2553

Car Rental
Budget Car and Truck Rental
Oak Ridge Mall
Oak Ridge, TN 37830
(615) 482-1324
(800) 527-0700

Taxi Service
Atomic City Yellow Cab
112 Randolph Rd.
Oak Ridge, TN 37830
(615) 483-4343

Cloud 9 Taxi and Delivery
P.O. Box 45
Hartford, TN 37853
(615) 483-3071

Other transportation services outside the immediate area are available.

The *Oak Ridge Convention Guide* was compiled by staff and interns of the Convention & Visitors Bureau; coordinated by Michelle McCarley; designed by Nancy Smith; photograph by Ruth Carey; typeset and printed by Sun Graphics.

COMPUTERS 101 (A Tutorial) (Part 4)

London, ENGLAND - Mark Knight

7 Programs and programming.

7.1 Machine code.

All computers actually run machine code programs all of the time, though this is often not appreciated. The processor cannot run anything but instructions from its own instruction set, and throughout the time it is switched on, simply repeats its instruction cycle, fetch, decode and execute, fetch, decode and execute. Machine code however, is just a sequence of nearly incomprehensible numbers, one after another, in the system memory. How is it that anybody can program using this code?

Simply put, machine code programmers don't program using machine code! They use a special code called ASSEMBLY LANGUAGE, which uses a set of symbols that are a little easier for humans to understand. For example, for the instruction to call a subroutine at memory address 45004, the symbols might be:

```
CALL    45004
```

... or, in another system:

```
JSR     45004
```

(Jump to SubRoutine). The instruction at the end of the subroutine, to return to the main routine, might be:

```
RET
```

...short for RETurn, or:

```
RTS
```

(ReTurn from Subroutine). It should be noted that these assembly language listings are still extremely difficult to understand for most of us, and the computer can't understand them at all. A special program, called an ASSEMBLER, converts the symbols to true machine code.

To really make the point, the following is a complete machine code program, written for the QL by the author of this text. It is a screensaver, and blanks off the QL screen if no key is pressed for a preset time, then restores it again when a key is pressed. Following it are some of the numbers that the assembler converts this listing into, written here in decimal. The first listing is the ASSEMBLY LANGUAGE, while the second is, in a sense, a fragment of the true MACHINE CODE listing. The assembly language looks like this:

```
* Screen saver, implemented as a Q-DOS JOB. © Mark Knight 1994.
        BRA.S   Start           Branch to actual JOB code.
        DS.B    4               Pad out to position JOB FLAG.
        DC.W    $4AFB           Mark file as JOB with standard flag.
        DC.W    11              Mark length of JOB name.
        DC.B    'ScreenSaver © 1994 Mark Knight.'
Start   MOVEQ   #$0B,D0         $0B=MT.PRIOR, vector to set priority.
        MOVEQ   #-1,D1          -1 in D1 selects this JOB.
        MOVEQ   #1,D2           Put priority in D2, in this case 1.
        TRAP    #1              Call MT.PRIOR to set priority.
        MOVEQ   #0,D0           Select MT.INF trap routine.
        TRAP    #1              To fetch system variables address.
```

```
            MOVE.L      A0,A4           Store the address in register A4.
MainInit    MOVEQ       #$13,D0         Select MT.RCLCK, read the clock.
            TRAP        #1              Call it, leaves time in D1.L.
            MOVE.L      D1,D4           Store time in register D4.
MainLoop    MOVE.L      A4,A0           Refetch system variables address.
            ADDA.L      #138,A0         Add 138 to base address.
            MOVE.W      (A0),D5         Store last key combination in D5.
            MOVEQ       #$08,D0         Select MT.SUSJB, suspend a JOB.
            MOVEQ       #-1,D1          This JOB again.
            MOVE.W      #14,D3          Put the length of suspension in D3.
            MOVE.L      #0,A1           No flag byte.
            TRAP        #1              Suspend it.
            MOVE.L      A4,A0           Refetch system variables address.
            ADDA.L      #138,A0         Add 138 to base address.
            MOVE.W      (A0),D1         Store last key combination in D1.
            SUB.W       D1,D5           Were they the same?
            BNE.S       MainInit        If not, loop back.
            MOVEQ       #$13,D0         Select MT.RCLCK, read the clock.
            TRAP        #1              Call it, leaves time in D1.L.
            SUB.L       D4,D1           New time - Old = Elapsed time in D1.
            SUB.L       #30,D1          Subtract 30 (secs) from elapsed time.
            BMI         MainLoop        If the answer is negative, loop back.
*
VideoOff    MOVE.B      #2,98403        Load 2 into the video mode register.
            MOVE.L      A4,A0           Refetch system variables address.
            ADDA.L      #138,A0         Add 138 to base address.
            MOVE.W      #-1,(A0)        Set an impossible key combination.
TestLoop    MOVE.L      A4,A0           Refetch system variables address.
            ADDA.L      #138,A0         Add 138 to base address.
            MOVE.W      (A0),D4         Store last key combination in D4.
            MOVEQ       #$08,D0         Select MT.SUSJB, suspend a JOB.
            MOVEQ       #-1,D1          This JOB again.
            MOVE.W      #14,D3          Put the length of suspension in D3.
            MOVE.L      #0,A1           No flag byte.
            TRAP        #1              Suspend it.
            MOVE.L      A4,A0           Refetch system variables address.
            ADDA.L      #138,A0         Add 138 to base address.
            MOVE.W      (A0),D0         Store last key combination in D0.
            SUB.W       D0,D4           Were they the same?
            BEQ         TestLoop        If so, loop back.
*
VideoOn     MOVE.L      A4,A0           Refetch system variables address.
            ADDA.L      #52,A0          Add 52 to system variables base address.
            MOVE.B      (A0),98403      Move correct byte to video register.
            BRA         MainInit        Loop back to main loop initialisation.
            END
```

The numbers output by the assembler, expressed as Words, look like this:

24616, 0, 0, 19195, 11, 21347, 29285, 25966, 21345, 30309, 29216, 32544, 12601, 14644, 8269 24946, 27424, 19310, 26983, 26740, 11776, 28683, 29439, 29697, 20033, 28672, 20033, 10312 28691, 20033, 10241, 8268, -11780, 0, 138, 14864, 28680, 29439, 13884, 14, 8828, 0, 0, 20033 8268, -11780, 0, 138, 12816, -26047, 26322, 28691, 20033, -28028, 1153, 0, 90, 27594, 5116 2, 1, -32669, 8268, -11780, 0, 138, 12476, -1, 8268, -11780, 0, 138, 14352, 28680, 29439 13884, 14, 8828, 0, 0, 20033, 8268, -11780, 0, 138, 12304, -26560, 26584, 8268, -11780, 0, 52 5072, 1, -32669, 24576, -136.

Yes, that's a program! The first version is comprehensible to a machine code programmer, while the second is closer to what the QL processor will understand. Incidentally, if you should have an assembler, and decide to type in and run the code, don't run it on a QL compatible system, only on a real QL. The code is compatible with Gold Card and Super Gold Card, but it could crash the system on anything other than a real QL. The dataspace should be set to zero using whatever method your assembler requires.

7.2 Interpreters.

An interpreter is a machine code program that allows a programmer to type, change, save and load symbols that are much easier for programmers to understand. The program will then sift through these symbols and call a set of standard routines to treat the symbols as instructions. An example is BASIC, or Beginners All-purpose Symbolic Instruction Code. (OK, OK, it should be BAPSIC really, but it doesn't read as well, does it?)

A SuperBASIC program, when loaded or typed in, is stored as a set of codes, with line numbers. A vastly simplified explanation will follow of how the interpreter program runs the fragment below:

```
2000 Total=Total+NewPrice 2010 PRINT "Total=";Total
```

So it is short, well you don't want to be reading until the middle of next week, and a long fragment would take that long, I assure you!

As the interpreter finishes the line before line 2000, it starts looking at this line. At the start, it skips the line number, then reads the code stored and finds Total in the program, recognising by the equals sign at the end that Total is a separate symbol. Finding this, it has to search through the NAME TABLE to see if it knows of a variable or routine with this number.

As Total is, in this case, a variable set up earlier in the program, the interpreter finds this, and then looks to see what is stored in the memory reserved for it.

Next the equals sign is noted. The interpreter then knows it must call the expression evaluator, a part of the system that works out arithmetic and does some other related operations. The equals sign means that Total is to be given a new value, based upon the result of some expression. The interpreter then fetches the next symbol, in this case, Total again. The name table has to be searched again, and the value stored in the memory is stored on the STACK. A stack is simply an area of memory that is used to store things temporarily, removing them in the reverse order to the order in which they are stored. The plus sign is fetched and stored elsewhere, and so the interpreter fetches the next token, NewPrice.

The same process occurs again, the interpreter searches the name table to see if it can find this symbol. NewPrice is found, and the number contained in it is put on the stack. The routine that adds the top two numbers on the stack is then called, and the two numbers are removed, added up, and the result is put on the stack instead. The result is then transferred from the stack to the storage space reserved for Total.

After this, the interpreter finds the end of the line, and so passes on to line 2010. The first token found after the line number is the PRINT token, and the search of the name table begins, to see if this is known. PRINT turns out to be a part of the interpreter, and so the interpreter calls the PRINT routine.

Finding the quotes, the PRINT routine notes that no channel is specified, as it is in:

```
2010 PRINT#2;"Total=";Total
```

When no channel is given, PRINT uses the value 1, and prints to that channel. The channel is looked up, and turns out to be a screen channel, so the screen driver is to be called. Before the screen driver is called, however, the characters in the string "Total=" are put on the stack, and then the number 6, in the form of a Word, is put on the stack to inform the screen driver how many characters to fetch from the stack and print.

The screen driver is then called. This is passed the location of the stack, so it can fetch the 6. Finding this, it starts to fetch the six characters, one at a time. As it finds each character, the dot pattern is looked up in the character set definition. Then the cursor position is looked up, and the screen is updated at the correct place to place that pattern, and the cursor position changed. The next character is then fetched until the end of the string is reached.

PRINT then finds the semicolon, and so knows that it need not change the cursor position before printing the next print item. The token for Total is sought in the name table again, and the number is transferred to the stack. Before it is printed, the number will be converted to a string using a complex machine code routine, and then printed just like the string "Total=" was. Returning from the PRINT routine and detecting the end of the line, the interpreter will find the start of the next line, if one exists.

Whew! Two simple lines of SuperBASIC, and yet remember that we have skipped hundreds of tiny details, and described others in a vague and possibly ambiguous fashion. Note also that at no time did the interpreter convert any part of the SuperBASIC program into machine code. An interpreter does not do that, it sifts through the program and uses it as data, to decide which interpreter routines to call. Much of the time is taken up in searching for things in tables, updating stores, and in working out which routines to call.

Interpreter programs are slower than machine code programs, and perhaps now you appreciate why. The SuperBASIC program is a file of instructions, just like a machine code program, but SuperBASIC needs to be repeatedly sifted through by the interpreter program in order to be executed.

The machine code programmer using assembly language is said to be using a LOW LEVEL LANGUAGE, because it operates at the machines own level of detail. The SuperBASIC interpreter, which provides a language far removed from this, hides the details of how the system works from the programmer, and as a result a language of this type is known as a HIGH LEVEL LANGUAGE.

7.3 Compilers.

A COMPILER is a relative of an Assembler. Instead of taking an assembly language listing and converting that to a machine code program, a compiler takes a different kind of program listing and converts that into machine code. Since the two are so closely related, the important things to describe here are the differences.

An assembler takes a low level language, the assembly language for that processor, and converts it directly to machine code. Each instruction in the assembly language program will be converted into precisely one machine code instruction for the processor. In the case of a compiler, however, a high level language is used, and each instruction in a program may be converted into one, two or many dozens of machine code instructions, depending upon the instruction and the precise circumstances.

Although some modern compilers are very good at taking a high level language listing and converting it into efficient machine code, they are still not as good as a human programmer in some respects. Compiled programs must use "libraries" of machine code routines that will work in many different programs, whereas a good human programmer will write code just that suits the program under construction.

A compiled program is almost always much faster than an interpreted program of the same type. Using SuperBASIC, the compilers available are SUPERCHARGE, Q-LIBERATOR and TURBO, and all of them produce machine code from a SuperBASIC program that is a vast improvement upon the interpreted version. The resulting programs load faster and run faster, and in the case of large programs, take up less memory, too.

Another advantage that some compilers can give is PORTABILITY. If written in machine code, then a program will normally only run on a single type of computer, but in many cases this is not true of a compiled program. A "C" program, for example, can be compiled on any system that has a standardised C compiler available for it. This is because the compilers will each turn out a different machine code program from the same C program listing, suiting the system the compiler is designed to run on.

Often programs require few alterations for a different machine, and occasionally none at all. This contrasts with machine code, where the entire program might have to be rewritten from scratch to run on a different system. In any case only a little of a complex machine code program is likely to be the same even if the new system has a similar processor.

Although the machine code turned out by a compiler is nowhere near as efficient as an assembler in the hands of a competent assembly language programmer, it has the advantage that it is very much easier to write. High level languages are designed to be easier than assembly language for a human being to learn. Compilers allow programmers to work much more quickly than assemblers, and depending upon the task at hand a really good programmer might choose an interpreter for quick, once only or small jobs, a compiler for more complex but not time-critical tasks, or an assembler where speed is of the essence or memory is short.

8 Operating systems and complete computer systems.

8.1 The early days - one supplier, one system.

In the early days of computers, each new system was a scientific experiment as well as a tool for the use of those involved. Each computer had its own operating system, completely different from other systems, and sometimes even different to other systems with identical hardware. When computers became practical as commercial products, this extreme situation had to change.

Commercial suppliers, like IBM and ICL, were only able to provide systems that were useful to their customers by standardising their operating systems within the company. It was, however, still the case that each supplier provided a unique system, and that selecting a supplier meant buying everything from that supplier.

Each IBM system might work with other IBM systems, but mixing ICL and IBM hardware was impossible, and software, too was often only available from one or two suppliers. This situation was only acceptable because of the size and complexity of early systems. A supplier, as well as providing a computer, would provide maintenance, software customisation and even operating staff for the computer. The price that was paid reflected this, and running costs were very much higher relative to the purchase cost of the system. Paying for IBM or Honeywell staff to help run your system pushed up the cost of ownership dramatically.

Eventually hardware became available to break the monopoly, as printers were made to connect to computers made by other manufacturers. Soon after, microcomputers were available, but still each manufacturer had their own operating system, and although you might have a choice of peripherals to connect to the system, choosing a computer fixed the sources of software for the life of that hardware.

8.2 The first portable systems - CP/M and UNIX.

When the C programming language was first popular in some academic circles, there were C compilers available for a number of different mainframe computers. As C was one of the first middle level languages, it was fast enough and flexible enough to be used to write an operating system. Previously, these had almost always been written in assembly language, and C opened the door for UNIX.

UNIX was first written in C, and was designed to allow the same operating system to be used on many different computer systems. UNIX was a multi-user system designed for large computers with many terminals connected, and so it was a multitasking system, too. The ability to run software on many different systems, as long as it was also written in C, made UNIX popular. A great deal of public domain and commercial software was written for UNIX, as it was little or no effort to write versions for different computers.

When the early desktop computers became available, a similar chaotic situation to the early days of mainframes arose. Each microcomputer had its own, unique and often arcane operating system, and users of different systems could only exchange data with difficulty. Programs were not portable at all.

CP/M was then supposed to end all this chaos, and for several years it did so very successfully. The letters stand for

Control Program for Microcomputers. The early CP/M systems ran on 8-bit 8080 processors, but the more powerful 8-bit Z80 and Z80A processors soon became the dominant force in CP/M computers. CP/M was not a multitasking system, but as the microcomputers of the day were not really powerful enough to support multitasking anyway this hardly mattered.

8.3 Multitasking - cooperative and preemptive multitasking.

There is much mention of multitasking in the world of computers now, as there always has been in the mainframe world. As microcomputers are now powerful enough to support multitasking, it is coming into use in almost all computer systems.

MULTITASKING is an illusion in most cases. The computer that is multitasking appears to be running more than one computer program at a time. This is not the same as having several programs loaded, but only one running, and having the ability to switch between them, which is called TASK SWITCHING. In a task switching system, one program runs, the rest are in store. In a multitasking system, more than one program may actually appear to be working at the same time.

The illusion of several programs running at a time is actually the processor switching between them every few microseconds or milliseconds. One program may run for a few dozen or a few hundred machine code instructions, then another, then the next, and so on, until the list of running programs is started again. As the user cannot detect the bursts of a few milliseconds individually, the computer appears to be running all of the programs together. In a single processor system, this will slow down as more programs are running.

The switching from one program to another is usually done by the operating system. There are two main types of multitasking, differing little to users but sometimes very different to programmers.

CO-OPERATIVE MULTITASKING requires programs to be written to give up the processor periodically as a result of the way they are written. A program written for such a system could hog all of the processing time, simply by not giving up the processor to the system. The operating system then decides which program is next to be given a "time-slice", and that program will run until it, too, gives up the processor.

Such a system is named for this friendly, co-operative method of programming. Writing for such systems in machine code can be difficult, though compilers will often simplify the job by taking care of the multitasking requirements without the programmer needing to know about them.

PREEMPTIVE MULTITASKING is used in systems where the operating system gives running programs no choice. Every few milliseconds the system forcibly suspends a running program and selects the next program to be run. This is a simpler system to program for, particularly in machine code, as it requires no co-operation from the running programs.

Preemptive multitasking systems are harder to write than co-operative systems, but they are often more secure, as bugs in a program do not allow it to take over the whole machine so easily. If a single program in a preemptive multitasking systems crashes, it is often possible to shut it down and continue using the system with the other programs running as if nothing had happened. With a co-operative multitasking system, it is often easier for one program to crash the whole system, other factors being equal.

One thing that is also common in multitasking systems, and is considered essential in mainframe systems, is HARDWARE MEMORY MANAGEMENT. This is actually a combination of hardware and software, allowing the operating system to protect areas of the system RAM from being used by the wrong program. So, a program loaded into memory could ask the operating system for 1Mb to store data, but if, due to a bug in its memory use routines, it tried to use memory outside this area, then it would be suspended and possibly closed down, with the system reporting the error.

Hardware memory management also prevents programs with bugs in from writing to the system variables, and protects the operating system itself. In most computers the operating system is itself in RAM, so it may well need

this protection if there is a bug-ridden program running. Hardware memory management is built into many modern processors, such as the MC68040 and PowerPC 601 chips used in advanced Apple Macintosh computers, and the 80486 and Pentium processors in IBM PC systems and compatibles.

The 68008 used in the QL does not feature hardware memory management, though it does have connections to allow an external memory management chip to be used. The QL has never had hardware memory management, and so no QL or QL compatible system that I am aware of uses hardware memory management.

8.4 The new breed - The Apple Macintosh and Microsoft Windows.

At about the time the QL was launched, the Apple Macintosh was also launched, using what was then the latest and best microprocessor available, the MC68000, running at 8Mhz, or half the speed of the one used in the Miracle Systems Gold Card. The Macintosh used a Windowing, Mouse driven and graphically-oriented operating system. It was later to become a co-operative multitasking operating system of remarkable speed and power considering the hardware it was running on, but the early system was merely capable of task switching, not proper multitasking.

Once the early Macintosh systems were replaced with systems having a decent amount of RAM, they began to be seen as the best of the mass-market business microcomputers. The Macintosh has almost always been seen this way, though the pricing policies of Apple held back the system for many years. Now, however, most Apple Macintosh systems are cheaper than typical IBM compatible systems of the same power. As the Macintosh is much easier to use and to set up than the PC system (even running Microsoft's Windows systems), it may well be about to achieve the success it deserves.

Microsoft Windows is really an attempt to bring the Macintosh ease of use to IBM compatible systems. The Windows system is similar in many ways to the Macintosh system to users, though it is somewhat slower on equivalent hardware. Everybody I have ever spoken to who has used both systems to any great extent prefers the Macintosh to the PC with Windows, but this has not stopped the runaway success of the Microsoft product. Millions of users are now using Windows 3.11, the latest version of the system, in spite of the problems inherent in setting up and maintaining a Windows system.

The current version of Windows is a kind of add-on to MS-DOS, the command-line driven operating system of the PC. Windows actually extends the MS-DOS system so much that it renders it unrecognisable, replacing the character based DOS screen with a graphical user interface used with a mouse. Windows 3.11 is a 16-bit, co-operative multitasking system, using the MS-DOS filing system and the standard IBM PC BIOS. It is slow compare to the Macintosh System Seven operating system, and very slow compared to Q-DOS, when running on roughly equivalent hardware.

8.5 Supercomputers and parallel operating systems.

Supercomputers almost always run one or other of the varieties of UNIX. One of the advantages of UNIX is that it is a fully 32-bit, preemptive multitasking system designed to handle multiple user environments and, in some varieties at least, machines with more than one processor. As some modern supercomputers are using larger numbers of processors, they are often using customised versions of UNIX.

There are serious problems with massively parallel computer systems, and these have not yet been fully solved. If a computer is to operate hundreds of processors, and divide a data processing task up so that these processors can all take part, then many things have to be done. One thing that has to be done is that each processor has to receive its own part of the job, both the instructions and data that it will need. The processors may also need to communicate somehow as the task progresses, to permit the supervising system to assemble the processed data.

This transmission of data and programs to and from hundreds of processors, each with its own RAM, creates a major communications bottleneck in all of the existing massively parallel systems. In some extreme cases, the system spends more time transmitting data and instructions about than it does actually running the software that it is intended to run, and obviously this is not desirable. Research continues on this subject, as it promises to speed up computers far more than any single advance so far.

As processors increase in speed, they are becoming harder to produce. If, instead of producing superfast processors, systems can be designed instead to use hundreds of thousands of cheap processors, then real computing speed is within reach. Some of the massively parallel systems peak at several hundred thousand million machine code instructions per second. Compare this with a Gold Card at approximately one million machine code instructions per second, or a Super Gold Card at a little over three million...

8.6 Home at last - Q-DOS.

OK, so what of the QL, running Q-DOS; what is so special about the system that we use?

If you have read this far, then you should understand this: Q-DOS is a fully 32-bit, preemptive multitasking operating system. It has a command line user interface, via SuperBASIC, and can present text and graphics on the screen in windows. The filing system, as supplied by Sinclair Research on the unexpanded QL, allows no subdirectories, but it can be extended to provide this facility. The Miracle Systems disk interfaces on Gold Card, Super Gold Card and some Trump Cards all provide the subdirectories on floppy disks that make large numbers of files easier to organise.

Compared with other modern computers, the QL is weak on graphics capabilities and a little slow in processing. Q-DOS, however, makes superb use of memory, and allows multitasking in a small amount of RAM, being perfectly usable within a 640k system, and extremely capable when running in the 2Mb on my Gold Card. Q-DOS does not impose the timing overheads of many other multitasking systems, and does not demand so much RAM, so it is faster than Microsoft Windows, for example, running on a PC that is theoretically more than twice as fast.

In order to run just one program sensibly, Microsoft Windows demands 4Mb of RAM and a hard disk. I frequently load five or six into my 2Mb Gold Card system and make use of them all, and this still leaves me over 700k free that Q-DOS can use as a disk cache. The Q-DOS disk cache is unusual, in that it is not of a fixed size, but makes use of any RAM that happens to be unused for other purposes. This means faster disk access when lots of memory is free, but it can slow down a lot when memory is very short. The automatic variation also means the user does not have to try to make decisions about how big the disk cache should be.

Two features of Q-DOS that make it special should be noted: Q-DOS uses much less memory than most multitasking systems, and it is easier to program than other multitasking systems. A third feature that affects performance is that it does not slow down as much as other systems when more than one program is running.

The fact that Q-DOS uses less memory manifests itself in several aspects of operation. First, Q-DOS itself is small, which enables it to fit into 48k of ROM complete with SuperBASIC as the interface language. This makes it cheap to produce a Q-DOS system. The RAM required to use Q-DOS is also less, so this, too, keeps the cost of a Q-DOS system down.

In spite of the small size, Q-DOS contains a large number of routines for managing the hardware, multitasking, input and output and other common programming tasks. This helps to keep programs compact, since programmers can often write code to call a Q-DOS routine, where on another system they would have to write their own code to do the same job. Not only are there many routines available, but the method of passing data to and from these routines is less complex, more compact and faster than the methods used by other multitasking systems.

Even if you are a user and not a programmer, the fact that Q-DOS is easier to program for will affect you. This is because you are less likely than with some other systems to find serious bugs in the software that you use, because the programmer has had less trouble writing it. Of course, it still doesn't make programming for a complex application easy, nothing can do that, but it is substantially easier than most systems and this results in fewer bugs.

The fact that Q-DOS does not need masses of RAM and the huge hard disk that most multitasking systems demand keeps the cost of the hardware down. My own QL uses a Gold Card and a colour monitor, with a PC-style keyboard. The system was, however, usable with the 640k of RAM that I had before the Gold Card arrived.

The computer that I have I could not afford to buy in one piece, but the way that the QL works allowed me to build it up a bit at a time. At each stage during my collection of extra RAM, disk drive, monitor, then add-on keyboard,

COMPUTERS 101 - (CONT'D)

new case with power supply and finally Gold Card, I had a perfectly usable system. Much of the credit for this must go to Q-DOS, for allowing sensible use of the memory and processing power that I had at each stage. Eventually I hope to add a hard disk, but even if I never do, I am happy with my QL and will probably continue to be so for years.

I hope that these articles have been informative. If any readers have further questions about any of the subjects discussed, please send them in and I will try to answer them in future issues.

(Editor's Note: As mentioned above, this is the final article in the series "COMPUTERS 101". This series has generated more positive comment than any other we have run in the last four years. If we all coax Mark, maybe we can get updates from time to time. Thanks Mark, for a job well done.)

Independent Producer's *Independent Producer's*

Mini Mart of Values

"The painless way to shop world wide "

Wood and Wind Computing
RR3 Box 92
Cornish, NH 03745 USA
Telephone: +1 603 675 2218

QLerk - A complete financial program for the QL

QLerk software (V3.21) with tutorial	$34	£23
QLerk manual	$38	£25
QLerk Software & Manual	$62	£41

DBEasy - A menu based database system

DBEasy software (V1.6)	$24	£16
DBEasy upgrade (give old version)	$7	£5

DBProgs - A toolkit of handy ARCHIVE procedures

DBProgs software (V1.8)	$18	£12
DBProgs update (give old version)	$7	£5

DBTutor - A general purpose learning program

DBTutor software (V1.5)	$12	£8

SEE our Full Page advert elsewhere in this issue

Albin Hessler Software

Im Zellfeld 25 D-72631 Aichtal
Tel + Fax +49 7127 56280

Software for the Pointer Environment

Cueshell (Ver 1.16) **$ 39.00 $ 63.00** DM 98
Desktop Program. Now with screen saver
CueDark especially designed for SMSQ

EASYPTR 3 Development System

Part 1 Basis Jobs	**$ 39.00**	**$ 63.00**	DM 98
Part 2 SBASIC Toolkit	**$ 19.50**	**$ 31.50**	DM 49
Part 3 Library Routines	**$ 19.50**	**$ 31.50**	DM 49

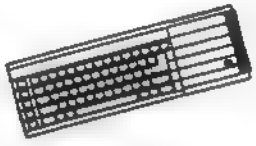
SERMouse **$ 29.00 $ 47.00** DM 73
QL serial mouse (works with SGC and SMSQ/E)
Ready for SER2 (specify BT or SUB-D9)

DISA 2 **$ 39.00 $ 63.00** DM 98
Interactive Disassembler. Many new features

Order through IQLR or directly from us
We speak English.
We accept eurocheques in DM or personal cheques in any EU currency drawn to any EU bank

Your Software/Hardware

Product(s) Could Be Advertised Here



Contact IQLR For Details

UPDATE! MAGAZINE
P.O. BOX 17
MEXICO, IN 46958 USA

The LARGEST magazine in the world supporting the Sinclair QL, Spectrum, TS2068, ZX81, and Cambridge Z88 Computers.

A quarterly magazine with issue year running from OCTOBER to JULY. We are NOW in our 8th year of publication.

To subscribe send a check made payable to UPDATE Magazine for the amount that is applicable to you.

USA & CANADA	**$20**
Rest of World	**£18 or DM40**

Cash, Checks, or Money Orders are acceptible.

How To Order

You may order direct from the supplier (prices listed include VAT where applicable and airmail post anywhere in the world. For your convience you may place your order through IQLR and take advantage of our ability to accept Personal or Bank cheques drawn on a UK bank or Euro cheques drawn in Pounds Sterling. You may also make payment using Postal (drawn in either US$ or Pounds Sterling), Personal or Bank cheques in US$ drawn on a US bank.

When ordering through IQLR, PLEASE make all cheques payable to IQLR and post your order to either our UK or North American office. We will process you order and the individual supplier will post it directly to you.

PLEASE NOTE: All ENQUIRES are to be made directly to the supplier NOT IQLR.

GETTING The Most Out of SBASIC

Duisburg, GERMANY - Jochen Merz

Converting special characters from DOS/TOS to SMSQ: It is very nice to have the flexible Level 3 Device Drivers of SMSQ - you can read and write DOS and TOS disks as if they were QDOS disks. There is absolutely no problem in converting bitmap graphics images or vector graphics, fonts etc. However, if you transfer text which contains special characters (foreign Umlauts, greek characters etc.) then run will find that VIEWing or COPYing the files will not convert the characters. Of course, it can't, as this would convert bytes in bitmap images as well, giving wrong results. But, we have SBASIC, haven't we. Some users seem to have problems in writing a few lines, but there are many ways of doing it: you can load in the whole file, modify it in memory and write it back - this is very memory consuming for long files, might not work on VERY long files but is very fast. As a compromise, you can do it in smaller chunks. That's more flexible, requires a bit more code, and is still fairly fast. The slowest method is handling byte by byte, but it is a very easy method, and doing it this way also gives a good example on how to write filters, that's why I choose this solution here.

First, what is a filter? It is a small program, which will get data from #0 and should send it to #1. It is up to the filter how it processes the data. So, ideal for translating characters.

Of course, there are many ways to translate characters. It could be done with a SELect clause, but this takes time. As we only have 256 possible character values, we can generate a table which holds the value in which a given character is to be translated.

But, first we give our job a name so that it appears in the job list identifying itself properly.

```
100 JOB_NAME "DOS/TOS to SMSQ"
110 :
```

Next, we prepare the table (an array) and fill it, so that any character translates into itself.

```
120 DIM tra%(255)
130 FOR char=0 TO 255:tra%(char)=char
```

Now let's deal with the characters which really have to be changed into a different character code. At the end of the program you will find a list of DATA statements. For every character to translate, there is a pair of data items. First, the "real" QL character, followed by the same character, but this time given in the character code which is used by DOS/TOS to represent this character. This means, if the program finds the DOS/TOS character value, it translates it into the QL character.

```
140 RESTORE
150 REPeat
160  IF EOF:EXIT
170  READ sms_char$,dos_char
175  tra%(dos_char)=CODE(sms_char$)
177 END REPeat
180 :
```

Here comes the real action. The job terminates if no bytes are coming into #0 anymore. It repeatedly reads characters from #0 and writes the translated character back to #1. Not too difficult!

```
190 REPeat
200  IF EOF(#0):QUIT
210  BGET#0,char
220  BPUT#1,tra%(char)
230 END REPeat
240 :
```

The data table contains translations for the German Umlauts only, but it is not a difficult task to add Italian, Swedish, French etc. characters, or even the greek symbols.

```
250 DATA 'ä',132
260 DATA 'ö',148
270 DATA 'ü',129
280 DATA 'ß',158
290 DATA 'Ä',142
300 DATA 'Ö',153
310 DATA 'Ü',154
```

Okay, that's it. You can save it, say, as ram1_DOS2SMS_bas (or to Harddisk etc., if you own it). Don't save it to RAM-Disk only, but if you have only one floppy disk drive and you're going to convert a file from a DOS disk, it is obvious that you can't have the program on your DOS data disk.

How do you use this filter now? In the same way as any other filter! You execute it, and follow its filename by two parameters - the file or device which is going to be #0 in the filter, and the file or device which is going to be #1 in the filter. Just try it. If we assume you have a textfile named "SAMPLE.TXT" on flp1_ (a DOS or TOS disk), then you could use

```
EX flp1_DOS2SMS_bas,'flp1_SAMPLE.TXT',#2
```

to view the contents of the file in your BASIC list window (provided, it is open, of course). If you wish to print it, use

```
EX flp1_DOS2SMS_bas,'flp1_SAMPLE.TXT',par
```

and if you wish to write it converted to your harddisk, then use

```
EX flp1_DOS2SMS_bas,'flp1_SAMPLE.TXT',win1_sample_txt
```

You have to use the quotes for the DOS filename as it contains a dot, and that's not a valid part of a QDOS/SMS name. If it does not contain any dot, like flp1_README, then you can forget about putting it in quotes.

More SBASIC hints: A very useful little trick: if you have the Menu Extension (Menu_rext) and QPTR (or a similar Toolkit) loaded, then there is an easy way of selecting a filename in a very comfortable way:

```
EXEP 'SBASIC';'HOT_STUFF FILE_SELECT$'
```

Of course, you can put it on a HOTKEY:

```
ERT HOT_THING ('f','SBASIC';'HOT_STUFF FILE_SELECT$')
```

It will pop up a full-sized file-select menu and stuff the name you select into the HOTKEY stuffer buffer. You can extract it by pressing ALT SPACE. Pretty useful if you want to select a file for, say text87, which has a file-select box inbuilt, but which is not as easy for handling files organised in various subdirectories on a harddisk, for example. You can provide the file_select$ function with all the allowed parameters (see QMENU manual), e.g. to select _T91 files only, use

```
ERT HOT_THING ('f','SBASIC';'HOT_STUFF FILE_SELECT$(,,,_t91)')
```

Another little hint: I saw that some users have put new procedures like LLIST etc. into their BOOT program. Yesterday, I had another call and it seems that a number of users simply don't know it. If you want to list your BASIC program to the printer, why not use

```
SAVE scr
```

or, if you want a form feed at the end,

```
SAVE parf
```

etc.? Of course, you can give ranges here as well:

```
SAVE parf,100 TO 500
```

will list only lines 100 to 500. One of the advantages of QDOS (and, of course, SMSQ) is its device independence. This means, that you can send and fetch a stream of data to/from any device, where device may also be a file. There is no reason why you should not be able to LOAD from a serial port either!

SBASIC debugging: Debugging under SBASIC is fairly easy: Especially with a history device, you can do more flexible, useful things. The following example is not a very useful program, it is just a few lines which is used for demostration.

The idea is, to provide a debug device to which all sorts of useful debugging information is written to. This may be a file, pipe, whatever you like. It is then up to you to put some useful print statements into your program, preferably at strategical points (i.e. entering a function, procedure, subroutine, list values of loop variables etc.) which might help you to trace the program flow. We can pass the device name in the parameter string to the SBASIC program, and if we don't give it, debug output is thrown away (i.e. written to the NUL device).

```
100 IF cmd$='':cmd$='nul'
110 OPEN#1,con
120 OUTLN#1,200,200,50,50
130 BORDER 1,4:CLS
```

We are now opening our debug channel.

```
140 deb=FOPEN(cmd$)
145 :
```

A small WHEN ERRor which will report any errors to our debug channel - might be useful! In case we don't want debugging, an error is reported and the program stops.

```
150 WHEN ERRor
160  PRINT#deb;'Error at line'!ERLIN:REPORT #deb,ERNUM
165  IF cmd$=='nul':REPORT ERNUM:STOP
170 END WHEN
175 :
180 PRINT #deb,'Entering loop'
```

Quick loop which will print some values and generate an error when it reaches 0.

```
190 FOR x%=-10 TO 10
200  PRINT #deb,'x%='&x%
210  PRINT x%;'*';x%;'=';x%*x%
220  PRINT x%;'/';x%;'=';x%/x%
230 END FOR x%
```

And that's it, more or less. Just wait for a keypress and die...

```
240 PRINT #deb,'Done...'
250 INPUT "Press ENTER"!wait$
260 QUIT
```

Save this program as, say RAM1_TEST_bas. Just try to execute it:

```
EX RAM1_test_bas
```

and it will print some numbers and stop, as a division by zero is still not possible. We can now use the parameter string, say, to send the debug reports to a printer:

```
EX RAM1_test_bas;par
```

GETTING The Most Out of SBASIC - (CONT'D)

And you can see how the program starts, stepping through the loop with an error report for x%=0 and it carries on until it reaches the end. This might be useful for short programs, but for a long program you might get endless paper listings. Not useful. You could write the result to a file and view it after the program has terminated:

EX ram1_test_bas;ram1_debug_output

But, there are more possibilities. For programs which take a long time to execute you might want to see where the program actually is (might be useful not only for debugging!). So, let's slow down our program deliberately to make it look like a heavily working program:

225 PAUSE #1,100

and save it again. Now we create another BASIC job which will allow us to watch the output of our test program:

SBASIC 11will create an SBASIC and make sure that the windows don't overlap with our test program. We need a report channel, therefore you enter the following line directly in the new little window:

OPEN#1,con_256x62a256x0:BORDER 1,4

Also, we set up a little spooler which will print every output from the test program into the new channel #1:

SPL pipe_debug,#1

Everything which will be sent to the pipe named debug will directly be spooled into the console channel. We can now execute the test program and see the effect. Of course, we have to tell it that it should send its output to the pipe:

EX ram1_test_bas;pipe_debug

You can see every step done in the test program being reported to your little #1 window. As soon as the test program dies, it closes its pipe-channel and the spool job gets an end of file, terminating itself as well. To avoid having to set up a spool job every time, all you need to do is to open an output channel to the same pipe which is never closed, e.g. type in

OPEN_NEW #4,pipe_debug

Lots of tricks, and here's a final one: if you have a really long program and you know that the program fails at some stage, which might take a long while - you don't know it, then all the methods mentioned above are not too useful, as even a file or a pipe will be filled up completely. That's where the history device comes into play: it remembers the most recent steps and "forgets" things which happened in the past. You define how much it can remember by adding a number to the history device name. Default is 1024 Bytes, which is usually more than enough. Let's try (but remove the PAUSE first from the program): **ex ram1_test_bas;history_debug**

Now, go into your main BASIC because it has some larger windows. You can now

VIEW history_debug

and you will see what you saw before, but in reverse order!!! This means the latest message (e.g. "Done...") is at the top, and if you look down you can see the execution of the program in the reverse order. This allows you to find the way back through tricky programs. You can press BREAK and view the history again (and again) and you will see that it is not emptied by looking at it (unlike the pipe - you read from it and it is gone). To demonstrate that the history forgets things, try

DELETE history_debug
ex ram1_test_bas;history_debug_30

and view the history again - only the last few steps are recorded.

GETTING The Most Out of SBASIC - (CONT'D)

I hope that this all did not sound too complicated - you should try it step by step on your SMSQ/E machine, of course. You will see how it works and how easy it all is. And if you don't need the features shown above for debugging, then I'm sure that you will find it useful for one of your (future) BASIC programs. You could use a small history to which a program which takes very long to complete can report its state of success from time to time. If you look at the history from another program you can find out how far the other program has got; if you don't want to know, you just don't look. If you do this trick with a pipe, you have to empty it otherwise it becomes full and the program which tries to print to it gets stuck.

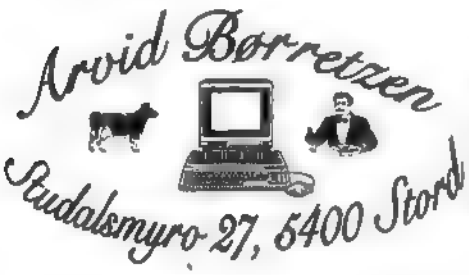
QDOS in a LABORATORY

Stord, NORWAY - Arvid Børretzen

In the beginning there was a little Casio calculator. Being a laboratory for food and water analysis, we needed many calculations. We did what everybody else did, used a calculator. Much of what was put into the calculator was the same every time. Our chemist, Ole Haukeland, bought a programmable Texas calculator to put formulas into. Now we did not need to enter the same things all the time, saving much typing and preventing many errors. But using this calculator meant you had to remember where to enter the numbers.

We read in the paper that there would be a demonstration of a new sensation, the Sinclair Spectrum. Ole and I attended this event , and we immediately understood that this little device could help us with our calculations. The 48 k Spectrum was ordered, and then a new hobby that almost has destroyed my old one, photography, began.

We soon found that the Spectrum could do more than calculating. Programs for storing results and making statistics were written. It was a big day when we could put the tape recorder away, and let the two microdrives do the storing. Much work is behind each byte of the stored data, so we have always backed up everything. We have never lost a single byte over the years.



The programs and the data files grew and the Spectrums 48 k was not so big any more. We needed more space. That same day I saw the introduction of the QL in an English computer magazine, I felt sure that this was the way to go. We ordered at once, but it took a long wait until the Norwegian distributor could deliver. A new era in our programming began. It was much easier to make good programs in SuperBasic. More and more functions in the lab were done on the QL. One computer was not enough. We bought more. At most 7 QLs were helping us keep the lab going.

We sent a big cheque away to Medic in England for disk drives. We only got one, and lost much money. Later we bought disk drives for all our machines. I think this was the most important Quantum Leap in our history of computing.

The QL did a good job, helped by Turbo and Qliberator, but some of them had problems with hanging up. Some 30 km away there was a little computer company named Futura (now Omega and not so little).

They managed to put QDOS into an Atari ST. This was the start of the emulators now sold by Jochen Merz in Germany. After a year or two, all our QLs moved to the loft and were replaced with Ataris. The reasons: Speed, memory, reliability and hard disks. But for us they are QLs. We only use them in QDOS mode (except for Calamus DTP).

For the time being we have 4 Ataris with hard disks and Brother laser printers. All are connected with FastNet from Qubbesoft. The net is very quick and has revolutionised the way we can use all our data effectively. All our backup is done over the network using my own program NorBack. Several lab devices (e.g. spectrophotometer, weights and titrator) are connected to the system via the serial port. The devices are manipulated from the system, and data fetching can be done very effectively. Programming for this is easy in SuperBasic.

QDOS in a Labratory - (CONT'D)

As we get new tasks and we find new ways to solve the tasks, the system grows and gets more sophisticated. Even without deeper knowledge of what is going on, you can get far with a soldering iron and a little trying. We have not experienced the great disaster yet. The system has become truly dynamic. We have a great advantage. We are two QDOS enthusiasts in the same place. New ideas come easier when you are more than one.

There are 80 food and water laboratories in Norway. 79 of them use a PC system and one (us) use a QDOS system. Why not join the 79? There are many reasons, but I think the main one is the joy QDOS gives. If we want a task done a little different, we change one or more programs. The 79 has to hope that the software house agrees to their wish, and maybe in a year there is a change??

I do not understand assembler programming. When I want something done that is too slow in compiled basic, I ask one of my friends if they can help me. Jochen Merz and Phil Borman have written many extensions in assembler, solving my problems.

I have tried to get a couple of drivers for windows changed to suite my needs. I asked for help from Microsoft. After waiting on the telephone for almost one day, absolutely nothing happened. The answer was that it was copyright protected material, and could not be changed. I wanted the printer driver for Text87 to behave a little differently. The source code came on a disk after a few days, and the problem was easily solved. You see the difference. In the QDOS community people seem to care a little for each other, in the PC community they seem to care for money.

At last a little about myself: 45 year veterinary. I have roots in USA and Norway. (My mother was born in Andaconda, Montana. Her father was French/Welsh. (Hi, Dilwyn)). My wife is Jorianne. We have two boys (15 and 18). They laugh at QDOS (and PC) and love AMIGA. I feel that today is more important than yesterday and tomorrow.

Ergon: c/o Davide Santachiara - Via Emilio De Marchi 2 42100 Reggio Emilia Italy - Phone/fax -39-522-300409

Ergon BBS +39 522 300509 - 22:00 to 04:30 CET ***FIDOnet*** 2:332/811 - ***INTERnet***: madmax@prix1.pr.infn.it

Ergon news: Ergon BBS!

Please note our ***new phone number for fax/voice calls: +39 522 300409.*** We can receive fax calls on this number ***ONLY*** from 10:00 to 20:00 local time (CET). Last year I promised to setup a BBS for better user support and finally here is ***Ergon BBS***. This is an authorised FIDOnet BBS (QBOX/QFAX based) carrying all International QL message areas. Call ***+39 522 300509*** from 22:00 to 04:30 local time (CET). Supported protocols: v21, v22, v22bis, v32, v32bis, ZyX 19.2; fax calls are supported too. Roughly 30 Mb of high quality and selectioned PD software are available for download and you can upgrade your Ergon program using your modem (full info about this on Ergon BBS).
A limited stock of ***Video-digitizers*** is now available. The Look & Show software has been improved and now works with GC/SGC.

Compatibility

All our programs run happily on (JM,JS,MGx,Minerva) Trump Card, Gold Card, Super Gold Card QLs, PC with QXL card and the Atari with QVME. SMSQ users please ensure you have v2.38 or later version. Our programs run with any graphic resolution (eg. VGA, SVGA), system variables can be at any position (full Minerva 2nd screen compatibles): ***future proof.*** Obviously they run happily with the Pointer Environment, PIE, PEX, PICE...

Prices, manuals

The price of our programs is all inclusive (airmail p&p, bank charges) and they are shown in different currencies. All the manuals are laser-printed and comprehensive tutorials with step-by-step examples have been added. If your manual is not laser-printed we highly recommend to get a new one. The upgrade price (see price-list) comprises p&p, the new manual and disk(s) with the latest version of the software.

Floppy Disk Utilities v1.21

Especially designed for the (Super) Gold Card user. This program fully supports DD, HD (1.44 Mb) and ED disks (3.2 Mb). Disk editor, single / multiple disk copier / formatter / verifier. It works also with a single disk drive. File recover, Search, Collect, Print sectors. You can also format disks with an extra 5% capacity (a bonus of 160 Kb with ED disks). The 40 pages manual includes 12 pages of tutorial. ***If you have a 3.2 Mb or 1.44 Mb disk drive this program is absolutely a MUST (now compatible with SMSQ + QVME).***

MasterBasic v1.44 plus 1

This is an useful development & debugging utility for the SuperBasic programmer. Reference edit PROCedures, FuNctions, variables, keywords, combination of tokens (eg. "FOR I=1 TO"), strings... All the reference operations are done in an easy, comfortable and fast way: all the referenced lines are put in a QPac 2 style menu: Choose with space/cursors the line to be edited or view it on a non-destructive window. MasterBasic can be also used with Minerva MultiBasics. The new 39 pages manual includes a five pages tutorial. ***Reviewers said:*** "Elegant and useful" (S.Goodwin QLW 4/93) "Easy to use and well thought out, even the more casual SB programmer should get a lot of use out of it" (R.Mellor QLW 6/92)

Q-Library Manager V2.30 NEW Version

Another useful program for the SuperBasic freak. This is a clever source code extractor / manipulator. Do you need to extract one or more routines from an existing SB program ? QLM will do it for you and will automatically extract routines called by selected ones. The new 30 pages manual includes a 5 pages tutorial. ***Reviewers said:*** "The package is extremely useful for SuperBasic authors" (R.Mellor QLW 8/91)

Open World V2.23

Convert GIF, IFF, TIF images into QL 4, 8 colours or monochrome screens. ***All screen resolutions now supported*** (VGA, SVGA...). Free a PC disk with utilities to convert QL screens into GIF images and to read QL disks on PCs, SUN, VAX. ***Reviewers said:*** "the price is very reasonable" (S.Goodwin IQLR 3/5)

DEA disassembler V5.21 plus 3

Interactive and powerful intelligent disassembler. The cheapest for QDOS. Commented assembly code (QDOS or SMS2 notation): Traps (QDOS, SMS, WMan, Ptr Gen...), System variables, Basic variables, Error keys. ***Automatic decoding*** even with very long files (usually no user intervention needed). ***Automatic Toolkit keyword extraction.*** All output codes are ready to be re-assembled (Hisoft, QMac, GST, Metacomco). DEA is supplied with a 56 pages manual (8 pages of tutorial). ***Reviewers said:*** "DEA provides all the utilities any machine code programmer could ask for" (Rich Mellor QLW Vol 3 Issue 5) "It's really useful and there's nothing else like it... I'm sure DEA will appeal many of the tinkerers and custom computer enthusiasts in Quanta" (S.Goodwin Quanta Vol 11/5)

Spectrum 48k/128k emulators

Again a lot of work has been done on our two best-sellers Spectrum emulators ZM/128 and ZM/hT. Major improvements are: 1) NEW: Flicker free screen on ZM/128 2) Switch freely to other tasks with the Pointer Environment 3) True ZX multitasking 4) Full 68000/20/30/40 Super Gold Card/QXL/QVME compatibles 5) Any screen resolution supported 6) AmigaQDOS compatibles 7) QXL's Net port supported.

ZM/128 plus 1 system

This package now comprises ***ZM/128 plus 1***, ***ZM/Accessories***, ***Backuppers***, 96 pages manual with tutorial. Some ZX PD games/utility are included. ZM/128 is a 48k/128k Spectrum emulator with Interface 1 emulation. ***Features***: Dynamic hardware selection, AY-3-8912 3-channel soundchip emulation, clever memory bank switching for fast 128k Basic emulation (fully usable even on GC), monitor, Z80 and ZTA file support, high compatibility (yet to find programs which do not run). ZM/Accessories allow you 1) to convert Disciple, Plus-D, MGT, Opus-Discovery disks into a suitable format 2) To read directly ZX tapes through the QL NETwork port 3) To convert Speculator files & snapshots into ZTA/Z80 format and viceversa. Backuppers allow you to transfer ZX tapes via network/serial port with Interface-1. ***Speed:*** usually >30% ZX speed on GC, 80% on SGC, >100% on QXL.

ZM/hT plus 2 system

This package comprises the full ZM/128 system plus the incredible ZM/hT Spectrum 48k emulator. ZM/hT is a state-of-the-art high technology Z80 dinamic compiler. ***This is the only Spectrum emulator for QDOS which runs at a reasonable speed on Standard QLs (eg. Trump Card) or at full ZX speed on a Gold Card*** (on the QXL it really flies, >300% Spectrum speed is normal). ***Reviewers said:*** "really spectacular... well worth the price" (E.Forenzi on IQLR Vol.3 Issue 1) "ZM/hT is very fast... well worth the extra cost.. an incredible feat of computer science" (S.Goodwin QLW 4/93) "awesome... even an 8 bit QL can run ZX software at reasonable speed ... most sophisticated..." (Last issue of Your Sinclair!).

SPEM professional hardware

System II cabinet: "Impressive. This is a white metal box with a flip-top lid, with fittings for your QL circuit board, hard disk, two floppies, two microdrives, power supply and all your interfaces." S.Goodwin QL World Volume 2 Issue 12. DIY mounting - you can easily fit a Qubide with 3.5" hard disk(s) and power supply.

Professional PC style Futura keyboard: The only keyboard which is 1000% QL compatible, no interfaces needed, no more membranes, easy to install, numeric key-pad, cursor keys are in the same place as on the QL keyboard. We use both them. Recommended.

SPEM digitizer: Connect any VHS source (video-camera, video-recorder...) to the QL and get 256x256 8 colours screen. The Look&Show software (included) now works with Gold Card & Super Gold Card.

Demo versions / Dealers

The ***ZM demo disk (V1.2)*** comprises PD versions of ZM/hT and ZM/128 plus some ZX games. The ***Ergon demo disk (V5.2)*** comprises PD versions of DEA, MBS, QLM, FDU and OWR. Please send 6 IRCs to get both. You can get all our software from Frank Davis in USA. Due to the closure of DJC our software cannot be bought anymore from Dilwyn Jones Computing.

New programs, upgrades and new manuals, hardware prices

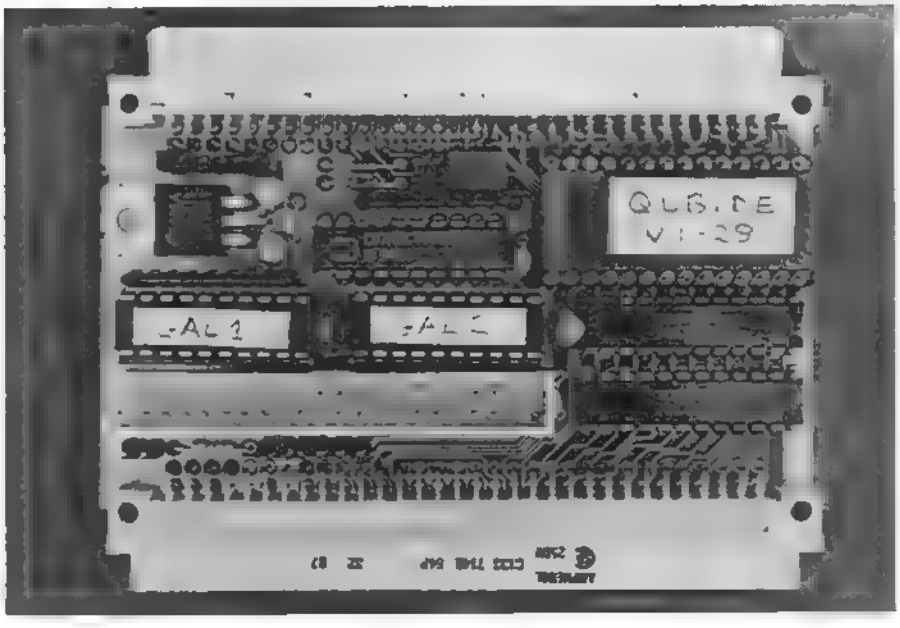
	Price full inclusive			***Program & Manual upgrade***				***Manual last revision (upgrade advisable)***
	USD	GBP	DEM	Pages	USD	GBP	DEM	
Spectrum emul. ZM/128 system	45	28	70	96	10	6	15	Jun '94
Spectrum emul. ZM/hT system	60	40	105	96	10	6	15	Jun '94
DEA intelligent disassembler	40	26	65	56	7	4	10	Aug '94
MasterBasic	35	22	55	40	6	3.5	9	Feb '94
Q-Library Manager	25	18	45	30	4	2	5	Aug '94
Floppy Disk Utilities	25	18	45	40	6	3.5	9	Jun '94
Open World	25	18	45	12	3	1.5	4	Jan '95
MusicManager	20	12	30	8	3	1.5	4	Feb '94
System II cabinet	100	65	175	p&p not included. Please enquire				
Futura professional keyboard	110	75	195	p&p not included. Please enquire				
SPEM digitizer & Look&Show	160	100	250	airmail p&p included				

USD: United States Dollar GBP: Great Britain Pound DEM: Deutsche Mark NEG: Nederland Guilder=DEM x 1.10

All the prices are inclusive of p&p, air-mail delivery, bank fees. Accepted forms of payment * Cheque in foreign currency * Eurocheque in ITL (1 GBP=2400 ITL) * Cash in ITL or foreign currency * International postal order in ITL * Direct bank transfer to Banca Popolare Dell'Emilia SWIFT BPMOIT012 Telex 510031 emipop - Sede Reggio E. Account no. 6533/73 D.Santachiara. Cheques payable to D.Santachiara. ***Discounts for group buy: enquire***

QUBIDE

QL AT/IDE Interface



QUBIDE brings the Sinclair QL into the 90's, It revolutionises the meaning of Mass Storage Media on the QL. With QUBIDE you are now able to connect modern AT/IDE Hard Drives to your Sinclair QL. A massive amount of storage space can now be made available for your programs and files. QUBIDE is fully compatible with Super Gold Card, Gold Card, Trump Card and most memory expansion systems for the QL, also Minerva and Hermes compatible.

To Order

Please make cheques payable to QUBBESoft P/D in Pounds Sterling.

QUBBESoft P/D
38, Brunwin Road, Rayne,
Braintree, Essex. CM7 5BU. UK
Tel: 0376 347852 Fax: 0376 331267

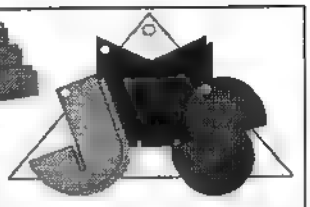
QUBIDE Price

£65.00p

fully inclusive of P&P (UK)
+5% (Europe), +10% (Rest of World)
Also includes 1 year Warranty

JOCHEN MERZ SOFTWARE

Im stillen Winkel 12 • 47169 Duisburg • Germany
☎ and Fax: 0203-501274 • Mailbox: 0203-591706



10 years Jochen Merz Software! I am supporting the QDOS scene now for 10 years, and that's a good reason to celebrate - 10 years of fun which resulted in a very close relationship to my customers, in many cases friendship (club meetings are tending to become a sort of "family" meeting. I would like to thank all users for their support.
Therefore, a special "birthday offer", valid until 1st of May 1995. Any new program or game purchase of any product found in the right column of this ad (no updates/upgrades) can be discounted:
4% if you buy 2 programs at the same time, 7% on 3 programs, 10% if you buy 4 or more programs at the same time!!!
Alternatively, any new program bought together with SMSQ/E will be discounted by 7%!!!

QL Hardware & Spares

FLP/RAM Level 2 device drivers for SuperQBoard		DM 56,–
FLP/RAM Level 2 device drivers for TrumpCard		DM 56,–
SER Mouse software driver		DM 40,–
SER Mouse Package (mouse, adaptor & driver)		DM 87,–
ZX8301		DM 19,90
ZX8302		DM 17,90
QL Keyboard membrane DM 28,–	2 membranes	DM 50,–
GoldCard		DM 299,–
SuperGoldCard		DM 699,–

QL-Emulators for ATARI

QVME – High-Res QL-Emulator for ATARI Mega STE and TT (up to 1024x800 pixels and more. PRICE CUT!!	DM 499,–
EXTENDED4-QL-Emulator for ATARI 260ST, 520ST/STF/STFM, 1040ST, Mega ST (but not STE!!!)	DM 289,–
400dpi Mouse (very high resolution)	DM 29,90

QXL (QL-Emulator for PC)

QXL 4MB DM 699,–	QXL 8MB DM 999,–

Have you already got QMON/JMON? Ideal for **SMSQ/E**, works on 68000, 68020, 30 & 40 (essential for QXL, SuperGoldCard and Hi-Speed ATARIs).
Remember: most products are not only improved but also upgraded to Config Level 2 - updating your software was never as easy as it is now: you configure your program once, and from then on MenuConfig does it for you whenever you get a new version!!!

Applications

QD	New Version	[V7.09]	DM 125,–
QBASIC	New Version	[V3.00]	DM 49,90
QDOS/SMS Reference Manual			DM 84,90
4 * Update sheets for Ref.Manual (incl. p&p)			DM 35,00
FIFI	New Version	[V3.00]	DM 49,90
QPTR		[V0.25]	DM 92,–
QSUP	New Version	[V3.02]	DM 79,90
QLQ		[V1.13]	DM 69,90
EPROM Manager	New Version	[V3.00]	DM 61,50
EasyPTR Part 1 DM 89,– Part 2 DM 49,– Part 3 DM 49,–			
LineDesign Version 2		[V2.06]	DM 249,–
FontPack for LineDesign (100 Fonts)			DM 199,–
HyperHelp for SuperBASIC		[V2.01]	DM 49,–
DISA		[V2]	DM 95,–
QMAKE	New Version	[V3.11]	DM 44,90
QMENU	New Version	[V6.00]	DM 39,90
typeset 93-ESC/P2 (Stylus driver for text87)			DM 69,90
QLiberator	Price Cut	[V3.36]	DM 139,–
WINED	New Product	[V1.08]	DM 49,90
DiskMate		[V4]	DM 69,–
QSPREAD		[V1.27]	DM 169,–
QMON/JMON		[V2.11]	DM 94,90
CueShell			DM 95,–
DataDesign V3			DM 149,–

Upgrades

HyperHelp Upgrade from V1	DM 6,–
FIFI Upgrade from V2	DM 6,–
FIFI Upgrade from V1	DM 16,–
QMAKE Upgrade from V2	DM 6,–
QBASIC Upgrade from V1	DM 6,–
QMenu Upgrade from any Version	DM 16,–
QSUP Upgrade from any Version	DM 16,–
EPROM Manager Upgrade from any Vers.	DM 16,–
QD Upgrade from V6	DM 16,–
LineDesign Upgr. from V1	DM 129,–
DISA V2 Upgrade from V1	DM 35,–
QMON/JMON Upgrade from V2.xx	DM 16,–
QMON/JMON Upgrade from QMON only	DM 32,90
QSpread Update	DM 16,–

SMSQ/E V1.47

More news on **SMSQ/E**: updates are free, just send master disk(s) & return postage (or use the free mailbox update service!
More new commands: HPUT, HGET, WGET, WPUT, LGET, LPUT, DISP_UPDATE (for QXL).
_ is accepted in DOS filenames now, the . for DOS/TOS filenames in SBASIC (no quotes needed anymore) Configurable initial screen size for ATARI
CTRL ALT SHIFT TAB gives a quick reset - many customers demanded for it.
Of course, lots of little improvements were made too: no disk on a GoldCard is MUCH faster, KEYROW problems fixed. ABC Keyboard supported. Revised OPEN_DIR (WREN, WDEL and WCOPY sometimes gave problems). And much more to come ...

	Feature	ATARI ST(E)/TT		(Super)GoldCard		QXL	
This exists NOW!	New Operating System	NEW	DM 199,–	NEW		already ex.	
	Multiple, fast BASICs	NEW	If you own the	NEW		already ex.	
	Flexible Level 3 Drivers	NEW	QJUMP-drivers	NEW	DM 199,–	already ex.	DM 199,–
	HD Disk-drive support (STE/TT)	NEW	Level C, D or E	already ex.		already ex.	
	TT Fast RAM support	NEW	already! else	impossible		impossible	
	Monochrome Screen-driver	NEW	DM 249,–	impossible		impossible	
Later	New Screen driver	NEW	+ DM 50,–	NEW	+ DM 50,–	NEW	+ DM 50,–
	"background" Disk/Harddisk	NEW	+ DM 50,–	impossible		NEW	+ DM 50,–
	BASIC-Development-Environment	NEW	+ DM 50,–	NEW	+ DM 50,–	NEW	+ DM 50,–
	Total price (when all is available)	with rebate,	DM 349,–	DM 299,–		DM 349,–	

As a special bonus (we know that many users own more than one system), we offer a full package which contains versions for all three systems for only 33% more. This also applies to the upgrades, e.g. DM 66,66 instead of DM 50,-, which, we think is very fair (it just covers extra disks and manuals).

Games

Diamonds	(Action)	DM 35,90
BrainSmasher	(Strategy)	DM 45,90
Arcanoid	(Action)	DM 35,90
Firebirds	(Action!)	DM 35,90
SuperGamesPack	(5 various games)	DM 90,–
QShang	(Strategy)	DM 45,90
MineField	(Strategy)	DM 39,90
The Oracle	(Strategy)	DM 49,90
BlackKnight	(Chess)	DM 119,90
DoubleBlock	(Tetris)	DM 42,90
Lonely Joker V2	(6 Card games)	DM 59,–
Lonely Joker V2 Upgrade from V1		DM 29,–
Pipes	New Game!	DM 29,–

If you require more information about any product, then please write for a free catalogue!

TERMS OF PAYMENT

Postage and package (Europe) DM 14,– (if total value of goods is up to DM 50,– then only DM 9,–). (Overseas) between DM 14,– (1 item) and DM 35,– (maximum). All prices incl. 15% V.A.T. (can be deducted for orders from non-EEC-countries). E&OE. Cheques in DM, £'s, Eurocheques and Credit Cards accepted.

EUROCARD. VISA

UPDATES

Updates of our software are usually free. The exception: major changes on a program (a new Version number before the '.'). Always send the master disk(s) to us, together with 4 international reply coupons for up to 5 discs or 8 IRC's for more. If you send updates together with a software order, then the return postage is covered by the wholesale postage. If a disk is faulty, add 1 IRC for a replacement. As the software changes from time to time, you may order a new manual together with the disc update. With upgrades, you automatically get a new manual.

Notes on QSI Speed Index Table

Berbenno, ITALY - Dr. Eros Forenzi

Below you will find the extensive results of recent tests using QSI Speed Index (available on most QL Bulletin Boards as public domain software). All but a few tests have been done by me using the English version of ARCHIVE 2.38 on an empty machine with just TK2 enabled.

If you do your own tests, use only English ARCHIVE 2.38, because other versions are slightly slower and test results could be different, especially on faster machines (eg. QXL). DO not use Xchange ARCHIVE, because it is much slower.

Overall speed is slightly slower if you do the test after loading Qpac2 desktop and buttons etc. Please note that a PC with DOS is MUCH slower if you run QSI PC in the same two machine conditions.

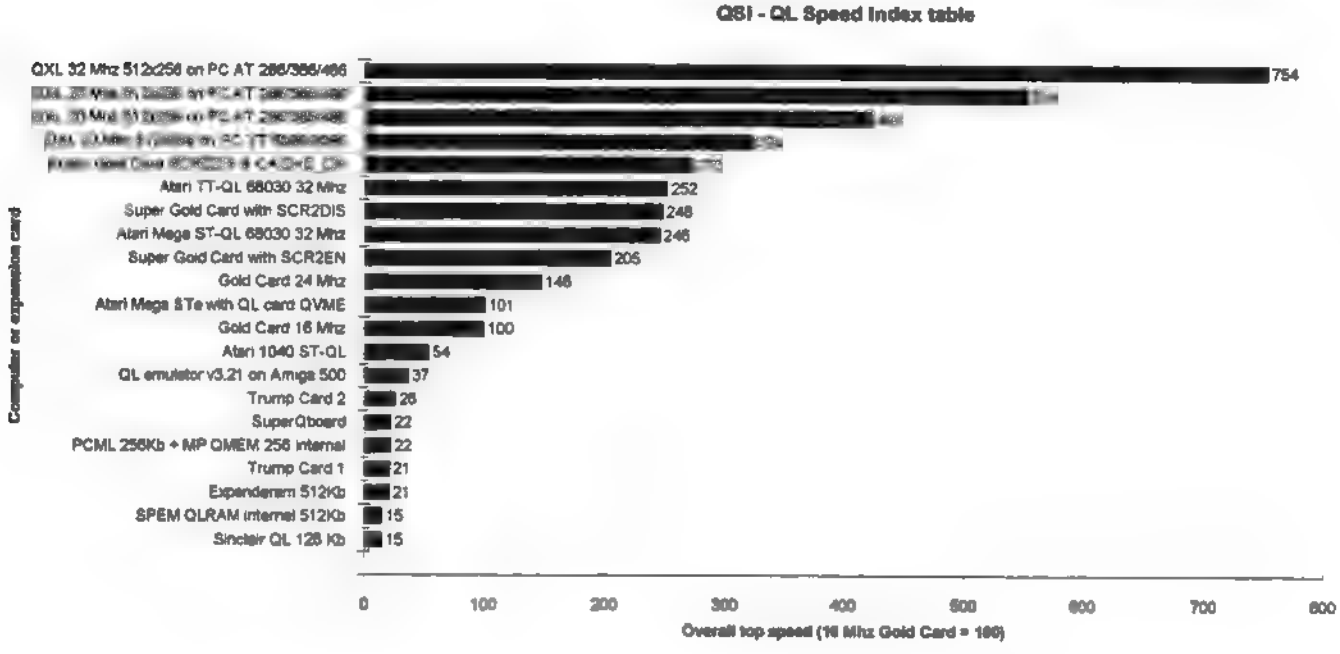
The Atari TT index is the courtesy of PROGS. The Operating system used was SMSQ/E, the 32Mhz QXL index is the courtesy of T. Godefroy.

The three Super Gold Card indexes are referred to different setups:

SCR2EN = second screen write enableed
SCR2DIS= second screen write disabled
CACHE_ON = cache enabled

All QXL indexes are referred to a QXLrunning in QL 512x256 screen resolution and with SMSQ version 2.10 or higher. Speed is roughly the same at least up to SMSQ v 2.39. The 25 and 32 Mhz QXL's are not official Miracle System products but are self-modified cards, either by replacing the 68040 with a faster variant or by overclocking the existing processor.

Speed of Atari-QL systems stays the same with higher resolution graphics. The QXL speed in EGA and VGA resolution is nearly equal to QL 512x256 mode. Speed in SVGA 800x600 mode is about 5% slower.



~ WOOD AND WIND SOFTWARE ~

Thank you IQLR for creating the Independent Producer's Mini Mart of Values. It provides a way for small dealers and developers to offer QL products world wide. Everyone benefits from this service both in terms of diversity and lower prices to users. Wood and Wind computing has been producing QL software for many years. We have reached a point where almost all of the energy running the business is produced from 36 solar panels. Although it may seem strange to some, ARCHIVE has been the platform of choice for software development. I believe that as long as QDOS and SMSQ are in use ARCHIVE will continue to be a very useful program. Why? Because computers are wonderful for storing and retrieving structured information (databases) and ARCHIVE is excellent as both a low level tool for dealing with the raw information and as a structured language that can serve as a platform for developing complex database applications [useful to everyone]. Everyone has a copy [available to everyone]. ARCHIVE works well in the QDOS multitasking environment on all possible mixtures of hardware, software, and ROMs [runable by everyone]. ARCHIVE has an interpreted language with built in editor so the code of any program is accessible to inspect and alter at any time [controllable by everyone]. ARCHIVE's slowness, a major drawback in developing complex ARCHIVE programs, has been overcome with the advent of the Gold Card, Super Gold Card, and the QXL. In a text based environment I believe that the 'Hot' Key Menus as developed for QLerk are just as easy to use as a mouse based system plus there is no need for any extra hardware or software to run the programs. In short, Wood and Wind Computing likes the openness of the ARCHIVE environment and plans to develop more QL software (in ARCHIVE and C) in the future. Listed below are our current software offerings. Long live QDOS/SMSQ. **See Mini Mart of Values for prices and ordering information.**

1) QLerk - A COMPLETE COMMON SENSE FINANCIAL PROGRAM

QLerk is a rock solid ARCHIVE based financial program for the home or small business. The only complete financial program available for the QL. The QLerk Program comes complete on a disk with a Tutorial Doc file. The most commonly needed features are covered by the Tutorial and those with simple needs would probably find this sufficient. There is also a complete indexed manual of 150 pages which is offered separately or together. Those with complex finances or a desire to know all the details should purchase the Program and Manual both. A minimum system would be a Trump card with 2 -720k drives. A Gold or SuperGold card or QXL work much faster. Not recommended for use in XCHANGE ARCHIVE. Special functions and reports can be created upon request. Reviewed in IQLR (vol 4 #4), QUANTA, and UPDATE.

2) DBEasy - A GENERAL DATABASE SYSTEM WORKING FROM MENUS

Create and use many different types of databases all from menus without ever having to know a single ARCHIVE command. Easily switch between them on the fly. Options include user definable displays, flexible ordering and selecting, print to printer or to file for import to QUILL, export for ABACUS or EASEL or ARCHIVE. This is a newly revised version (V1.6). Menus have been improved and a new 'Capture' option has been added so you can easily bring any ARCHIVE database or Export file into the DBEasy environment. Works well on any QL with disk drives in ARCHIVE or XCHANGE ARCHIVE. Those already owning DBEasy are offered a special upgrade price. Reviewed in QL World. A PC version of DBEasy is available for those who also work in a PC environment and have PC ARCHIVE. Please inqure if interested

3) DBProgs - A TOOLKIT OF HANDY ARCHIVE PROCEDURES

A collection of useful ARCHIVE programs that can be used on any ARCHIVE database plus many useful ARCHIVE tips. This is a newly revised version (V1.8). Included are DR (do a directory), SCAN (view and print any database), FREQ (construct a frequency table of fields of any database), SPLIT (split any database into 2 databases), JOIN (Join 2 similar databases into 1), REFIELD (Add or change field names and positions in any database), REPLACE (find and replace text in any database either by field or everywhere), MATCHER (locate duplicate or similar records in any database), QUERY (query any ARCHIVE database using logical constructions), FORMED (create and save forms using fixed text and database field values. View and print with font options - normal,bold, underline,condensed,wide). Great for special labels. These procedures work well on any QL with disk drives in ARCHIVE or XCHANGE ARCHIVE. Those already owning DBProgs are offered a special upgrade price.

4) DBTutor - A GENERAL LEARNING PROGRAM

Want to learn something? Your QL can help. DBTutor allows you to set up special learning databases of information you want to learn and will then drill you in many different ways. Allows for single or multi-line questions and answers. Prints out quizzes and will keep your score. Works well on any QL with disk drives in ARCHIVE or XCHANGE ARCHIVE. The interface is not up to QLerk or DBEasy standards but it works well. Reviewed in IQLR (vol 1 #5).

DJC

DILWYN JONES COMPUTING

41 BRO EMRYS, TAL-Y-BONT, BANGOR, GWYNEDD, LL57 3YT,
GREAT BRITAIN. TEL+FAX (+44)0248-354023

Address Book & Label Printer	£15 00
AMD Airplan	£10 00
Banter	£25 00
BASIC Reporter	£10 00
Bible text disks	£20.00
Cocktails Waiter	£10 00
Convert-PCX	£10.00
Cricket Secretary	£12 00
Cue-Shell	£40 00
Data Design 3	£60 00
Data Design API	£20.00
DBEasy	£15 00
DBProgs	£15 00
DEA disassembler	£26 00
Deskjet-A5	£12.00
DEV Manager	£15 00
Disa 2	£40 00
Discover	£20.00
DJToolkit	£10 00
Easyptr 3 part 1	£40 50
Easyptr 3 part 2	£20 00
Easyptr 3 part 3	£20 00
Filemaster	£12 00
Files 2	£12.00
Five Game Pack	£12 50
Flashback	£25.00
Flashback Special Edition	£40 00
Fleet Tactical Command	£39 95
Flightdeck	£15.00
Floppy Disk Utilities	£18.00
4Matter & Locksmithe	£23.50
FTC Data Printer	£9.95
Fugitive	£9.95
Genealogist 3	£60 00
Genealogist 2nd Edition	£30.00
Genealogist 2 to 3 upgrade	£33 00
Genealogist 128k (budget version)	£12.00
Grey Wolf	£12.50
Home Budget	£20.00
Image-D	£10 00
Image Processor	£15.00
IQLR current issue	£4.25
Line Design 2	£100 00
Locksmithe	£14.95
Masterbasic	£22.00
MDV Toolchest	£14.95
MegaToolkit (disk)	£25.00
MegaToolkit (eprom)	£40 00
Micro Process Controller	£64.50
MPC software	£9.95
Toolkit for use with Process Controller	
Multi Discover	£30.00
Music Manager	£12.00
Network Prover	£4.00
OPD Interchange	£15.00

Open Golf	£12 50
Open World	£18 00
Page Designer 3	£40.00
Painter	£25.00
P.A.Y.E. Master (UK only)	£30.00
PD2 Clipart	£10.00
Pfb2Pff	£60 00
PFData	£20 00
PFList	£20 00
Picturemaster	£15 00
Picturemaster Plus	£20.00
Pointer Environment Idiot's Guide	£2.00
Printermaster	£20 00
Proforma	£100 00
Proforma Font Pack	£80 00
Publishers Pack	£199 00
QIndex	£20 00
QL-PC Fileserver	£24 50
QLiberator 3.36	£50 00
QLiberator, budget version	£25 00
QLibrary Manager	£18 00
QLOAD and QREF	£15 00
QPAC1	£19 95
QPAC2	£39 95
QPower regulator	£24 95
QRactal	£20.00
QReview issues 1-4 (each, UK price)	£2.00
QTOP	£29.95
QTYP2	£29.95
Question Master	£10.00
Quick Mandelbrot	£15.00
Quick Posters	£10.00
Quiz Master	£10 00
Ramdisk	£2.00
Return To Eden	£17.50
Rob Roy Pack	£10 00
R.P.M.	£15.00
Scanned Clipart 1	£10.00
Scanned Clipart 2	£10.00
Screen Compression	£10 00
Screen Dazzler	£15.00
Screen Economiser	£10 00
Screen Snatcher	£10 00
S_Edit editor	£20 00
SerMouse (+£2.50 postage)	£40 00
Mouse Mat for Sermouse	£2 50
Sidewinder Plus (screen dumps)	£24.95
Sidewriter (sideways text printing)	£15.00
Slowgold	£5 00
Solitaire	£15.00
Solvit Plus 2	£20.00
Solvit Dictionaries	£10 00
Spectrum Emulator Zm/128	£28 00

Spectrum Emulator Zm/HT	£40 00
Speedscreen (disk/mdv)	£15.00
Speedscreen(Eprom+disk/mdv)	£30 00
Spellbound	£30.00
Spellbound Special Edition	£50.00
Squidgy Round The World	£12.50
SToQL	£12 50
Super Disk Index	£12.00
Super Disk Labeller	£10.00
Taskmaster	£25.00
Taskmaster hard disk upgrade	£5.00
Text87 Plus 4	£79 00
2488 Text87 24 pin drivers	£29.00
Typeset 94 Text 87 HP drivers	£29.00
Fountext98+Founted89 (Text 87)	£39.00
Textidy	£15.00
Text 'N' Graphix	£20.00
The Clipart	£12.00
The Gopher	£12.00
3D Terrain	£12.50
Toolkit 2 (disk/mdv)	£19.95
Toolkit 2 (EPROM)	£24 95
Trans24	£10.00
Video Titles	£5.00
Vision Mixer 1	£10 00
Vision Mixer Plus	£22 50
Winback	£25.00

FLOPPY DISKS & LABELS

FLOPPY DISKS DSDD 3.5"	£0.40
FLOPPY DISKS DSHD 3.5"	£0.70
ED DISKS only	£2.00
DISK LABELS 3.5" roll 100	£2 00
DISK LABELS on tractor feed	£2 50
ADDRESS LABELS roll 100	£2 00
Microdrive cartridges	£2.50

Add £2.50 postage for floppy disks, or £0.50 postage if only ordering labels.

MAGAZINES

QReview magazine UK per issue	£2 00
QReview (Europe)	£2.50
QReview (other countries)	£3.75
IQLR per issue	£4.25

CATALOGUE

Call or write for a FREE copy of our full 24 page QL catalogue.

DISCOUNTS (SOFTWARE ONLY) 5% FOR 2 PROGRAMS, 10% FOR 3 OR MORE!

TERMS AND CONDITIONS. All prices shown in UK Pounds Sterling. Software sent post free to U.K., overseas add £1.00 per program (maximum £3.00). Bulky items - see above for postage. PAYMENT - Make cheques or Eurocheques (in Pounds Sterling, drawn on UK branch of bank or building society) payable to "Dilwyn Jones Computing". Payment by credit card (Visa/Access/Mastercard/Eurocard) also accepted. Minimum order is now £5.00 due to bank charges). Goods, whether individually identifiable or not, remain the property of Dilwyn Jones Computing until payment in full for them has been received by DJC. Under credit card company rules, we can only send goods paid for by credit card to the cardholder's registered address.

Credit Cards
VISA
ACCESS
M'CARD
EUROCARD
all accepted!

DJC DILWYN JONES COMPUTING

41 BRO EMRYS, TAL-Y-BONT, BANGOR, GWYNEDD, LL57 3YT, GREAT BRITAIN.
TEL+FAX (+44)0248-354023

SOLVIT-PLUS 2

NEW!

A brand new word games program. Solve anagrams, scramble-searches, wildcard searches to find words with missing characters or groups of letters (? and * wildcards), palindromes, backward words, even a simple spelling checker! Very simple to use and above all great fun - ideal for crossword and word game lovers! Multi language dictionaries (English, American English, French, German, Spanish, Italian, Dutch and Welsh dictionaries (which are also available separately as plain text file wordlists if you'd like to use them with your own software, for example). Requires minimum of 512k expanded memory, available on disk only (supplied on 4 disks, including wordlists). £20.00

DEV MANAGER

NEW!

A useful little utility for helping with getting software to run from hard disk or directories on floppy disk, even difficult software like Quill. Compile a list of DEV etc settings for all of your programs, saves having to type in endless DEV or SUB commands. Pointer driven program. Compatible with Miracle, Qubide and Falkenburg hard disks. Expanded memory required, disk only. £15.00

DEA

Powerful machine code disassembler program by Ergon Development. Intelligent, able to disassemble machine code extension, knows QDOS system calls etc. £26.00

FDU

Floppy disk utilities, disk editor, can help to recover corrupt files etc, on DD/HD/ED disks. £18.00

LINE DESIGN

It was used to create this advert on a QL and HP Deskjet 500 - need we say more? Now you can produce good looking mixed text and graphics on your QL too! Expanded memory required (Gold Card or other large expansion preferred). Compatible with virtually all printers, scaleable fonts and graphics, pointer driven.

LINE DESIGN 2	£100.00
PUBLISHERS PACK (including Text87Plus 4)	£199.00
PROFORMA FONT PACK (high quality extra fonts)	£80.00
PROFORMA interface for C68 users	£100.00
PFLIST text listing utility	£20.00
PFDATA data listinn utility	£20.00
PFB2PFF Adobe type 1 font conversion utility	£60.00

QPAC2

If you want to use pointer environment, QPAC2 is a MUST. This is the complete system, including the PE itself, Files, Jobs, Sysdef, Hotkeys, Buttons, Things, all those terms you've read and heard about. If you are serious about multitaskng and your QL in general, you need QPAC2 - makes task switching and other chores which are tedious on a standard QL a doddle! Requires expanded memory, disk only. £39.95

PE IDIOT'S GUIDE by Norman Dunbar, free with QPAC2 or by itself for only £2.00. A simple introduction to pointer environment.

QPAC1 - A small package of useful little accessory program for use with or without QPAC2, includes on screen calculator, calendar, typewriter, alarm, etc. £19.95

CLIPART

Supplied at £2.00 per disk, subject to usual £5.00 minimum order value.

RELIGIOUS (7 disks)	£14.00
CHRISTMAS (2 disks)	£4.00
FANTASY etc (2 disks)	£4.00
MINICLIPS (1 disk) (small detailed pictures)	£2.00
COMPUCLIPS (1 disk) (computer-related clipart)	£2.00
AI FILES (3 disks) (for use with Line Design only)	£6.00

PAGE DESIGNER 3

Easy to use pointer driven desktop publishing program for QL. Printing not to the high standard of Line Design, but less than half the price! Includes free disk of clipart, a disk of fonts, various printer drivers and even font editors! Ideal intro to DTP on the QL. Expanded memory, disk only. £40.00

A Trip Down Memory Lane

Santa Clara, California, USA - Jim Hunkins

Diversity is a good word to use when talking about QL users. This has been well demonstrated by the series of articles run in IQLR describing how different QLers use their systems. To further illustrate this impression, and with the hopes that others will find similarities in their use of QL systems, I have agreed to contribute my own 'how I use my computer' story. This article will attempt a chronological type flow. I have structured it to follow my exposure to computers from the first. Hopefully, this approach will prove to be both interesting and perhaps a bit entertaining.

Before going any further, a brief introduction is in order. A few of you may recognize my name either from some of the articles I have written or from communication on the different networks. I have been an avid Sinclair user from the early days of the ZX81. My profession, electrical engineering, pays for my hobby. I also have a second degree in computer science. I wish I could say that I have both degrees because I am brilliant, but the truth is that I couldn't make up my mind as to what career path to take. Instead I ended up working harder and longer to cover all my choosen options, resulting in double degrees. Professionally, I specialize in high speed digital design. When I have the opportunity, I also mix a bit of systems and software engineering in. However, I consider my software work to be a hobby, not my profession.

As with most of us, I like to think that I don't fit any particular stereotype. I started out to be an architect but switched to photography (everything from retail to teaching to shooting to darkroom). Then I discovered something that I could both make a living at and enjoy, computers.

In the beginning...

Okay, enough background information. It is time to take a brief look at my several years of exposure to computers. It's amazing just how far things have come since my first hands-on contact with a 'computer'.

My initial exposure was with one of the cheap wire jobs from Radio Shack back in the 60's. The kit was one that you wired a couple and/or gates together and programmed by connecting a few other wires to specific points. Technically, it was a computer, barely. But it was a start.

In 1972 at the University of Colorado I was introduced to a mainframe. In those days programs were always flow charted to avoid mistakes. [A flow chart is a picture of the program flow drawn as boxes for each step and lines connecting them.] To physically enter and run your program, you had the pleasure of waiting in line for a key punch machine. There you typed in short, simple program lines, one per card. You then took the stack of cards to the main campus where you waited in line again. Finally, at the head of the line, you handed your stack to a complete stranger who would rather be any where but in a noisy room at three in the morning. Then your stack was put in a tray with thousands of other cards. If you were lucky, no one else's cards would jam or cause the system to go into an infinite loop before your cards got a chance to jam or send the system into an infinite loop. Obviously, this was an experience to be avoided. Believe me, I got very good at flow charting.

Despite the 'klunkiness' of the early systems, I persisted and made it through a semester of Fortran (an early version that had just added +- functions). Even though intrigued, computers were not yet ready for me (or maybe I wasn't ready for computers). That was to be my only personal exposure to programming for several more years.

In the late 70's things started happening. I picked up a Sinclair ZX81 computer kit and learned to solder. I then bought another kit and successfully completed building it. Out came the cheap black and white TV and with a little patience (and cable positioning), I had a blank screen with a prompt. The thrill of entering my first line (something like: 10 PRINT "hello, jim") and seeing the TV say "hello, jim"... Well, I believe that you all understand. Lets just say that it was a small improvement over my mainframe experiences.

At that time I had a camera store which was doing reasonably well. After entering what seemed like hundreds of programs from magazines and books and writing some simple games myself, I proceeded to try my hand at business software. The first result was an accounts payable program that ran on my little ZX81 with 2K ROM and 16K (expanded?) memory.

A Trip Down Memory Lane - (CONT'D)

I used that program to help me keep track of what bills were due when and what discounts would occur if they were paid off by a certain date. Looking back now, the program was crude by today's standards. But for back then, being on a cheap computer, and being my first attempt at a major program, it was actually pretty good (modesty is obviously not one of virtues).

Next came the Timex Sinclair TS2068. On to color screens, better games, high resolution word processors, and bank switching (since it used a Z80 processor, it had to do this to access more than 64K of memory.) When I picked up this computer, I had a choice between it and the Commodore 64. The TS2068 was much more capable than the Commodore. It took Commodore nearly two years to catch up. Unfortunately, the month after I bought the computer, Timex dropped out of the computer business. So much for my dreams of a gigantic market place! It was on the 2068 that I really started playing around with assembly language and starting learning how operating systems worked. One of the nice things about Sinclair computers has always been the abundance of helpful people and the amount of published information on how things worked. I definitely took advantage of that.

The TS2068 was also my introduction to a modem. I remember my first 300 baud modem. It wasn't fast, but the magic of being able to communicate with others across the phone line without speaking was, well, magical. Never mind that much of my early experiences were spent 'talking' with a friend only a few blocks away. Who needs to physically speak anyway?

Enter the QL

Time marched on and now, the QL appeared. When the QL was first introduced, I had just gone back to school and could not afford the $600 price tag. It sounded really awesome, but as most of us are painfully aware of, money is not always just lying around to be spent. So I kept using and learning from my TS2068.

When they discontinued the QL, the news was met with mixed feelings. I was really surprised and dismayed to see it go. On the other hand, the resulting price slashing brought it down into an afordable price range. I promptly took part of my student loan (read starvation diet) and sent away for one.

After working with terminals on a slow and cumbersome mainframe at school along a few of the early, awkward PCs, the compact black box with a built-in keyboard was a bit too different. So of course I immediately plugged it into my television and started computing (or I should say playing around). The instruction manual would have to come later.

I have spent the years since that day learning, playing, and having a great time. As with the TS2068, I started buying books on it whenever I could. I even had a college bookstore 400 miles away (I lived in South Dakota then and was probably the only one in the entire state aware of the QL) special order a book from an European publisher. I grabbed hold of the early QDOS reference manual and quickly got lost in it (literally lost, most of it was beyond me!).

As I delved into my black box, without realizing it at the time, I was learning about a rather advanced operating system. This came in very handy a few semesters later in school when I started studying the UNIX operating system. For those of you not exposed to UNIX (be happy), it is a very large and cumbersome (but powerful) multi-tasking operating system used on main frames and work stations.

The similarities between QDOS and UNIX are truly amazing. The biggest difference is that, while QDOS and UNIX have many similar capabilities, QDOS does everything with much less code and overhead. When the instructors starting talking in class about daughter jobs, memory protection, and other fun stuff, almost everything mapped directly to something contained in QDOS. It sure made those semesters easier.

And then came my AI (artificial intelligence) class. Armed with my copy of the Metacomco LISP language interpreter, I did most of my assignments on my QL and then transferred them to the main frame. For my final project that semester, we had to solve a pyramid stacking puzzle using either LISP or Prolog. I wrote my program in LISP on my QL and then wrote a SuperBASIC program that graphically animated the LISP results. I then packed my QL and microdrives off to school to do a bit of showing off (the QL of course, not me). I also had several engineering and assembly code programs that I was able to write on my QL. Some of them I handed in as is, while others I translated back to Fortran or 'C', depending on the class. With my modem, I was able to remotely login to

our main frame from home to do some of my Fortran assignments. This was a lifesaver when the school labs shut down for the night and I still had work to do, especially in my compiler class.

Obviously, my Sinclair machines have done me well over the years. Since leaving school, I have pretty well retired the other machines due to lack of time. With the one exception of a couple of years ago when I ran my Christmas lights on my TS2068 with a controller card design that was originally printed in one of the early magazines. Come to think of it, I might pick up the I2C interface board and use my QL to do it this year. Hmmm...

I have picked up one other Sinclair computer since leaving school, the Cambridge Z88. This has been handy as I take it on the road with me. I can work on programs for my QL, write letters, or stuff like this article when not at my desk (poolside is a favorite place of mine). Then I bring my Z88 back in and transfer what I was working on back to the QL. Oh yes, a tip for traveling on the plane. I always get extra attention with my Z88 since they think that I am a business traveler working away. I don't let them in on the truth that I am just having fun.

My QL and I - today and tomorrow

What do I do with my machine now? Well, it gets a bit harder to pin it down. At work I am primarily an electrical engineer. At home I like to think of myself as a software engineer. My trusty QL systems are my main hobby. Not only are they advanced and ever growing, but they are also different enough from the 'PC world' at work that it is a complete change of pace from the job.

I like to play with different programs for example. Yes, the word is play. I dig into them to try to get a 'feel' for how they work. Understanding a program in such a way gives me an insight while trouble shooting a program's bug or unexpected results (there never has and never will be a bug free program). I also use the insight when thinking of different ways to do things for some of my own programs.

Speaking of which, I do a fair amount of programming, both in SuperBASIC (SBasic) and 'C'. With the help of the C68 'C' compiler (a big thanks go to Dave Walker and all his cohorts for their porting and expansion work on this), I have been improving my 'C' skills. Most of the programs I do are small things that I need or just to see if I can make something work. My latest effort which some of you might have seen is the QL BBS Message Reader (okay, it isn't so small), which I have made available as freeware to get more people active on the nets.

I am currently tackling programming for the pointer environment. My first use will be a port of a simple file transfer utility (Z88 <-> QL) to the pointer environment. Then I plan to develop a full featured graphics oriented 'difference' program (never have been happy with what is available, both in the QL and the PC worlds).

And the list of projects that I want to work on is never ending (refer to my project list at the end of this article). If anyone has any suggestions or wants to tackle any of these, feel free to. I have enough to do as it is to last me for years.

My hardware has grown to support my ever growing needs. Originally, my QL gained a Trump Card with expanded memory and a floppy disk interface. Now it has a Gold card. Along the way, I added the Hermes coprocessor (serial ports work like a champ now) and the Minerva ROM (as do all of us, I love my toys and take good card of them!).

I plan to eventually expand it even further (how about a EvenMoreSuper Gold card with the 68060 - hint, hint). However, at this time I have taken to using my QXL card in a PC box for most of my work. I find its speed and memory is awesome. Until last week, I was running the QXL from DOS and Windows (forgive the PC talk) and am now running it under OS2. I find that most of my serious work is on the QXL, while the PC half of the system is an excellent games machine (yes, even I take breaks from all this).

My modem has grown from the original 300 baud unit used with the TS2068 to a 14400 unit which I run at 9600 from the QL and at 14400 from the QXL. I find my self on line a lot these days. I used to think that the faster the modem, the less time I would spend on line.

But access is so easy and there are so many things to get into and people to meet, there just isn't enough time for everything. My main locations are QBOX-USA (access to US QLers and the European FIDOnet connection),

A Trip Down Memory Lane - (CONT'D)

Compuserve (access to other QLers around the world and an unbelievable variety of special interest forums), and now some Internet work (haven't even begun to tap that yet). By the way, it is very interesting, after having talked with people across the nets and elsewhere for years, to actually meet several of them. I had the pleasure of doing this last spring at the 'Miracle in Newport' QL show. I am really looking forward to next year's event!

As for the future... my QL systems will be with me for some time. They have and will continue to be a source of enjoyment, relaxation, education, frustration, and accomplishment. As you can see from my project list, I don't expect boredom to be a problem. One of my longer term and exotic projects will involve AI. Can you imagine my QXL card working with me on my PC as my counter personality. You may have heard people saying that their computer had developed a personality of its own. I plan to eventually have my QXL do just that, develop its own personality. But as I said, don't hold your breath. It will definitely be a LONG term project.

Thanks!

I do wish to express my gratitude to each individual who has put time into the international QL community. This includes everyone from the weekend word processor user to the game player to the up-all-night tinkerer to the publishers to the hardware and software houses. You have all managed to take a machine and its operating system that were exceptional in their own right, and make them grow and live, in ways that I suspect no one could have imagined. I have and will continue to enjoy being part of this community for many more years.

Hope to see you all on line soon!

Personal System Summary (this is only a partial list, exclusion usually means that I haven't come up with the money yet!)

Hardware List:

System 1	System 2	Misc
QL	QXL (vHBA)	Intel 144/144e modem
Gold Card v2.32	1-60M hard drive	Canon BJ-200 printer
Hermes v2.18	1-40M hard drive	
Minerva v1.82		
2 - 3 1/2" drives (dual density)		
serial mouse		

Project Wish List:

- network communication script language
- graphical difference program
- QXL front end for Windows/OS2
- Scanner and CD interface for QXL
- enhanced graphics display
- hard disk utilities with compressed backup
- QXL as specialty PC coprocessor
- (AI personality - neural nets)

Please Take Note

Most of our UK advertisers list the addition of a 1 in their telephone number. Please make a note of this change in your records. Both telephone numbers (the old and the new) are valid at this time, but in April the old numbers will no longer be accepted.

Please also take note of the change of address and telephone/fax number for Miracle Systems in their adverts in this issue.

QITALY Club - The Italian QL Users' Club

Keeping the QL alive!

If you are an Italian QL User, please consider joining us.
You'll be in contact with almost everything the QL world has to offer, in Italy and abroad. We are not an alternative to Clubs like Quanta or magazines like IQLR and QReview. We strongly recommend to join/subscribe all of them too!

What we offer

QITALY Magazine - electronic magazine
MONDO QL - paper magazine
QITALY BBS - Fidonet BBS
QITALY Program Line - wide range of PD
Hardware repairs
Free support, help and advice

QITALY Magazine comes on 720Kb disk and is PD.
Each issue contains articles, news, programs and screenshots taken from PD and commercial software. Most of QITALY Magazine is viewable onscreen directly from the disk (just BOOT it).
QITALY Magazine is in Italian, with short info in English.

MONDO QL is an A4 paper magazine out at irregular intervals (about 2 to 4 times a year). It is all in Italian and contains more info on particular topics.

QITALY BBS is our own Fidonet node. We carry all the major QL international echomail message areas and a few Italian message areas, all about the QL. Nearly 20-30 QL Users here in Italy regularly connect to QITALY BBS. There are also many PD programs available for free download (about 500 programs as of Dec '94).
QITALY BBS is available on +39 342 590451 24h a day on Saturday and Sunday; 5:00 PM - 7:00 AM GMT, Monday to Friday.
You can connect with any modem up to ZyXEL 19200 bps - soon V.34

Rates

QITALY Club association (1 year): 35.000 ITL *
(You get Mondo QL paper magazine, unrestricted access to the BBS, hardware repairs, etc.)

QITALY Magazine subscription (4 issues): 25.000 ITL *
(You will get just the disk magazine)

* or rough equivalent in your local currency. Countries outside EEC please add 5.000 ITL for each of the above. You can pay by Cheque, Eurocheque, International Postal Order or Cash. Please send Cash by recorded delivery only.

How to contact us

Foreign contact section: EROS FORENZI - Via Valeriana,44
23010 Berbenno (SO) ITALY

phone: Eros Forenzi +39 342 590450 (5:00 PM - 10:00 PM, GMT)
FAX: Eros Forenzi +39 342 590451 (5:00 PM - 7:00 AM, GMT)
e-mail/Internet Eros.Forenzi@f123.n331.z2.fidonet.org

Do not hesitate to call us for more info!

QL Forever!

THE PE - an idiot's guide!

Newmachar, Aberdeen, SCOTLAND - Norman Dunbar

1. THE POINTER ENVIRONMENT

This article is intended as a user's guide to the Pointer Environment. Some astute readers may already have noticed the words an idiot's guide - please don't be offended - the idiot is me !

It is not a manual on how to use the Pointer Environment, nor will it teach you how to write programs. It is my intention to have this booklet included with all my Pointer programs, thus removing the need to duplicate information in all of their manuals.

If the program you have just purchased is your first Pointer driven one, then please read this manual in order to gain a better understanding of any technical terms used in the manual for your new program. If you are a seasoned user of Pointer programs, then this manual might still prove useful, but you will probably be aware of most of the information contained in it.

The Pointer Environment, sometimes called the Extended Environment, is a method of allowing the QL to have things such as non-destructive windows, the use of mice (mouses ?), menus etc. The Pointer Environment was designed and written by Tony Tebby and Jonathan Oakley.

Many programs are now being written to use the PE, as it will be known from now on, especially programs written on the Continent by people like Jochen Merz etc. Dilwyn Jones is trying to get all his authors to convert their programs to use the PE, I know, I am one of them.

This manual can be used by the 'PE unaware' so that a better understanding of what these 'new fangled' programs are all about. This manual does not cover any particular program in detail, but gives a general overview - once you can follow what is going on with one program, it should (note, should) be easier to follow the rest.

Although this outline does not relate to any particular program, various screen dumps etc will be shown to illustrate a point or two. These screen shots are all taken from a single program, but the issues being discussed are valid for most, if not all, PE programs.

2. HOW TO TELL IF YOU HAVE THE PE OR NOT

If you buy a program from Dilwyn Jones Computing's adverts or software catalogue, you will find mentioned somewhere that the program in question requires the PE to run. In most (all ?) of these cases, you will be supplied with the required parts of the PE that the program needs.

Other advertisers also state whether the PE is required and if it is supplied or not.

There are so many program that require the PE these days and that seem to supply the same files on each disc, that some people might get confused and think that these files that keep turning up, **PTR_GEN, WMAN, HOT_REXT, MENU_REXT** etc, are public domain - they most certainly are not. These files are Copyrighted and for each program sold, a royalty payment has to be made to the authors.

3. WHAT FILES MAKE UP THE PE ?

The basic PE is considered to be made up of three main files, **PTR_GEN, WMAN** and **HOT_REXT**.

3.1 PTR_GEN

PTR_GEN is the first file that must be loaded into your QL, if not, an error will occur when one of the other two programs are loaded. PTR_GEN is loaded by using the LRESPR command from Toolkit 2 or by the original RESPR, LBYTES and CALL commands for those without Toolkit 2. Other toolkits may have their own versions of these commands. The outcome is that PTR_GEN installs itself into your QL and not a lot happens !

PTR_GEN is the part of the PE that is responsible for handling the **POINTER** that you see on screen. This normally takes the form of an arrow head pointing to the top left corner of the screen, but some programs change the pointer to something more suitable for their own needs. Pointers are discussed in more detail below.

PTR_GEN extends the handling of CON_ channels within your QL - but you as the user don't need to know about this nor do you have to do anything. Easy stuff this, so far.

The first thing that you will notice after PTR_GEN has been loaded is that when you use CTRL and C to switch between a number of tasks that are running in your QL, the screen never needs to be refreshed !

For example, imagine that you have loaded a copy of EDITOR by Digital Precision, and SuperBasic, as usual, is running. You are typing something in to the EDITOR and need to save it, but you cannot remember which files are on the appropriate drive. Normally you would use CTRL C to switch back to SuperBasic, type in something like DIR FLP1_ and press ENTER, check out the list of files then CTRL C back into EDITOR and press the key combination to redraw the screen display which has been written over by the list of files.

What now happens is that when you CTRL C back to SuperBasic, the original contents of the SuperBasic screen are automatically refreshed for you, and the EDITOR screen vanishes (or maybe part of it does). Having DIR'd to get up the list of files on the device, when you switch back into EDITOR, the screen changes back once more to the EDITOR display without you having to do anything. (In fact, I cannot remember what the screen refresh keys are in EDITOR these days !!!!) So, we have an advantage already.

Another 'bonus' of PTR_GEN is that it likes to keep the screen tidy. Some peple may not consider it a bonus, but I do. QDOS, the QL's Operating System, is a multi-tasking operating system and allows more than one program to run in the QL at the same time. This is an illusion, but for simplicity we will assume that they all run together.

When a program opens up a window to use for its display, it usually overwrites whatever was on the screen before and sometimes you can have two or more programs all merrily writing to the screen at once, lovely mess !

PTR_GEN only allows the program that is 'at the top of the pile' to write to the screen. All the others are suspended until they are on the 'top of the pile'. This seems to destroy the QL's multi-tasking abilities but more of this later.

When you start up the QL, SuperBasic is running and nothing else. Any program that you run can therefore use any part of the screen that it likes with no problems. If, however, you now EXEC another program, it too may require a bit of the screen to write to, your EDITOR for example. If the new program opens a window which overlaps part or all of the SuperBasic windows, then SuperBasic is prevented from writing to the screen.

This procedure is carried out for the next program, which may stop the EDITOR writing to the screen, and so on. Any program which does not require to write to the screen does not stop running so multi-tasking is still going on. I don't think that there are very many (useful) programs that don't need to use the screen from time to time !

I have recently heard about a new 'version' of PTR_GEN that allows the parts of a programs windows that are not covered by other program's windows to be written to. I have no experience of this program yet. (This is not a Tony Tebby version)

PTR_GEN has a couple of procedures built in to it. These being **CKEYON** and **CKEYOFF** which stand for Cursor KEY ON and Cursor KEY OFF respectively. These commands allow you to select whether the arrow keys on your keyboard will move the pointer or not. CKEYON makes the pointer move when an arrow key is pressed, CKEYOFF means that a mouse has to be used.

Summary of PTR_GEN

PTR_GEN controls the pointer that you see on the screen, handles the extended CON_ drivers and keeps the display tidy when multi-tasking. It is the first file that has to be loaded when the PE is to be used.

3.2 WMAN

WMAN is the second file to be loaded in order to use the PE. It supplies additional features over that supplied by PTR_GEN. WMAN is an acronym for (of) **Window MAN**ager, and that is exactly what it does.

WMAN supplies a number of utility routines that are used by the vast majority of programs that have been written to use the PE. The routines handle such things as setting up a window, menus, loose items etc and that is basically it. Don't worry about these technical terms as they are all explained later on.

There is not really a lot more to say about WMAN !

3.3 HOT_REXT

HOT_REXT is the third part of the PE and this one has numerous facilities that are available to the 'ordinary user'. HOT_REXT provides your QL with the HOTKEY system. Hotkeys can be set up to cause some action to be carried out when they are pressed.

On all QLs there are the five function keys, F1 to F5. In the early days of the QL, these could only be used by detecting them using KEYROW or INKEY$ within a program, but they couldn't really be programmed to do anything useful for some time. This was not very much use.

Toolkit 2 provided a command called **ALTKEY**, which allowed almost any key on the keyboard to be programmed with a useful command or function to be carried out when that key was pressed in conjunction with the ALT key. For example, ALT and C could be used to run the C68 compiler by typing the following :-

ALTKEY 'C', 'EX CC;"-v -TMPram1_ -oram1_test_exe test_c"', ''

Which, without getting too complicated simply means that whenever the user holds down the ALT key then presses the C key, the command in quotes, would be typed in to the keyboard queue and then ENTER would be simulated. This would have the effect of carrying out the command. This was much more useful as fairly large commands could be shortened to a couple of keys pressed at the same time.

The HOTKEY System, as provided by HOT_REXT, is a **THING** (see later on !) and provides the user with a fairly large number of procedures and functions that can be used to load, exec, wake (more technical stuff) programs, or other THINGs etc.

Most of the commands and functions start with the word HOT_, for example, HOT_STOP, HOT_GO, HOT_LIST.

Summary of HOT_REXT

HOT_REXT provides the HOTKEY system which allows commands to be programmed into certain keys on the keyboard. These commands can be used to initiate or switch to jobs or tasks running in the QL. The HOTKEY system is based on a THING.

4. THINGS

I am led to believe that THINGS are so named because Tony Tebby and Jonathan Oakley couldn't think of another name for them so thay ended up discussing 'those things that you wrote yesterday' etc and the name stuck - THINGS they are then. Things are implemented to allow tasks to communicate with each other. For example, Digital Precision's TURBO compiler uses a THING like method to pass information from the PARSER task to the CODEGEN task.

The PARSER runs and if there are no errors, leaves information hanging about in the QL's memory for the CODEGEN task to read and use to produce the compiled code. How on earth does the second task know where the first task left the information ?

TURBO picks up its information from a system variable, I believe, and so the CODEGEN task knows where the PARSER left the information. THINGS work in a similar manner, but there is a slight difference.

THINGS can have all sorts of uses, they can be executable things, extension things and all sorts of other things (!)

Probably the most used example of THINGS would be Jochen Merz's QMENU package. This is implemented as a series of extension THINGS, ie, THINGS that act like an extension. This basically means that as far as SuperBasic is concerned, they are just another toolkit, but with subtle differences.

The advantage of a THING is that they can be used by almost any program in the QL. Using QMENU as an example, I can write a SuperBasic program that uses them, a C68 program that uses them or an assembler program that uses them. They all use the same copy of the extensions and can find them in the system without the user knowing where they live.

For those who may have used QMENU but were unaware of it, the file is called MENU_REXT and is responsible for many PE programs having a very similar method of selecting a filename to load or save etc. There are other extensions included in **MENU_REXT**, but I will not labour the point here.

So that is the basic background to the PE. Next, I will dive in and try to explain what all those technical terms that get mentioned in manuals are about.

5. THE TECHNICALITIES

After you obtain your first PE program, you probably notice things (no relation to the above !) in the manual that puzzle you. Words like **OUTLINE, MENUS, POINTER, LOOSE ITEMS**, etc etc. What are they and how do you use them ? Read on.

5.1 POINTERs, HITs and DOs

A POINTER is simply a small picture, normally an arrow head, that moves around the screen when you move the mouse or press one or other of the arrow keys. You could also call it a cursor, but that would be slightly inaccurate. The POINTER points. (Does a cursor curse ?)

As you move the pointer around the screen, you might notice that in certain places, a rectangle appears and disappears - you have just found a LOOSE ITEM, but read about that below.

The pointer basically shows you where something is likely to happen.

A **HIT** is when you move the pointer to a specific place on the screen and either press the SPACE key on the QL or press on the LEFT mouse button. A **DO** is when you press ENTER or the RIGHT mouse button.

In good old fashioned QL mode, in the days before PE, when you wanted the QL to DO something, you invariably had the press the ENTER key. Type in a line of code and nothing would happen until you pressed ENTER, once that was done, the line of code would be carried out. So ENTER became the DOing key.

Some of the most often seen pointers are :-

showing, from the left :

the normal pointer, the 'window is locked and you can't use it' pointer, the 'there is no window defined here' pointer, the 'needs mode 8' pointer, the 'this window needs some keyboard input' pointer, the 'can't use this window' pointer, the 'move' pointer and finally, the 'size' pointer.

The normal pointer shows up most often and for most programs, will be the only one that you may see, although the programmer can change it to something else.

The padlock means that the window that the pointer is over is partially buried and so the job owning the window is currently locked from use. You are able to use this window if you HIT or DO while the pointer is over the window in question.

The window request pointer shows up when you move the pointer to an area of the screen that is not used by any of the jobs running in the QL at that time.

The mode 8 (and not shown above, the mode 4) pointer appears when the area of the screen where the pointer is belongs to a window that is not in the same mode as the rest of the screen currently showing. The pointer shows a 4 or an 8 depending upon the mode requirements of the program owning the window. If you choose to use a program whose pointer shows that a change of mode is required then the mode will change automatically. This will cause all windows in the 'other' mode to vanish. Moving the pointer over the screen will show them up when it turns to the 'mode' pointer.

The keyboard required pointer shows up when the window you have the pointer over belongs to a program that is currently waiting for you to type something in.

The no entry pointer is used when you have the pointer over a program's window and that program is currently not interested in anything else other than what it is doing. It could be in the middle of a long calculation and may have its cursor turned off. There is no way that you can get at this program yet.

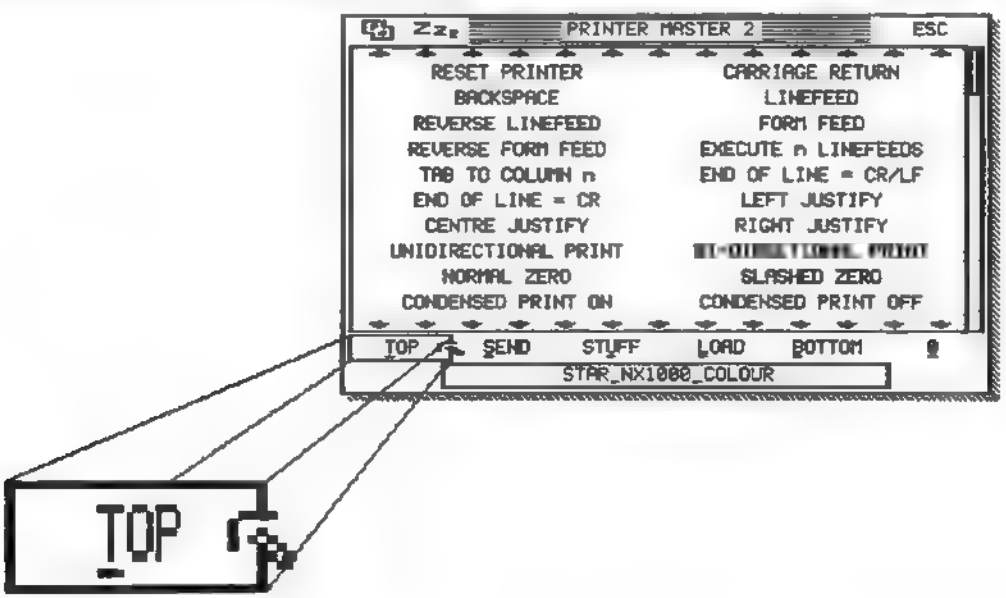
The move pointer is used when you press CTRL and F4 together or HIT/DO on the MOVE loose item (see below). You now move the pointer to where you want the window to be positioned and then HIT or DO it.

Finally, the size pointer is used when you wish to redraw the window bigger or smaller than currently shown. This is, of course, possible only if the program allows it.

5.2 LOOSE ITEMS

These are simply words or pictures on the screen that have been programmed to carry out some function when HIT or DO'd. They are called LOOSE ITEMS because they can appear almost anywhere on the screen and are not tied down to any one particular place. This doesn't mean that you, the user, can move them about, it is more the author of the program who decides on their location, so he can move them about. By the time you get them, they are stuck down firmly !

The figure below shows a screen from a program called Printer Master 2 and highlights a single loose item.



As you can see in the enlarged portion, the loose item has a border around it - this shows that the pointer has been positioned within the area defined to be the boundary of this loose item. The border appears so that the user knows that something can be done here.

Also shown is the **SELECTION KEY**, which is the letter that has been underlined. Pressing this letter, here it is a 'T', will carry out the same action as will be done when the loose item is HIT or DO'd.

Other loose items can be seen in the main part of the figure, one at the top left looks similar to one of the pointers shown above. This is quite normal. A loose item can have a picture on it as well or some text depending upon which makes most sense to the programmer. There are a number of loose items along the bottom as well, these being, **TOP, SEND, STUFF, LOAD & BOTTOM**. At the top there is **MOVE, SLEEP** and **ESC**.

A HIT on a loose item may carry out some action and a DO some other action, but this need not always be the case. In some programs a HIT and a DO have the same effect, others do not. This is entirely up to the programmer when the program is being written.

Loose items normally come with a word or picture on them that gives you some idea of what they do. In many programs, a loose item will have one of the letters in the word underlined as a further way of carrying out the required action.

These selection keys are a way to avoid having to always move the pointer over the loose item in order to get something done. This is very handy for people who don't have a mouse. Simply leave the pointer where it is and press the action key and things will happen.

5.3 'STANDARD' LOOSE ITEMS

Many programs appear to have the same set of loose items somewhere on their screens. There may be programs where only some of them are present and others which use them all. These are the 'standard' loose items, **MOVE, SLEEP, SIZE** and **WAKE**.

5.3.1 MOVE

The **MOVE** loose item looks like this : 

When this loose item is HIT/DO'd the pointer turns to the move pointer. Placing the pointer in a new position on the screen then HITting or DOing it causes the whole window for the appropriate program to be moved to the new position.

The move loose item can be chosen using the mouse or keyboard to position the normal pointer over it, it will show a border when the pointer is in the correct place, then HIT or DO it. It can also be selected by pressing CTRL and then F4 (hold down CTRL, press F4, then release both).

Regardless of which method is used, when the pointer is moved to a new position and HIT/DO'd, this will be the new position of the pointer when the window is moved. If there is no room to fit the whole window around the new pointer position, the display will be adjusted to stay in range.

5.3.2 SLEEP

The **SLEEP** loose item looks like this : 

If you are a user of the program QPAC2, then HITting or DOing this loose item will put the program to sleep as a button in the button frame. If QPAC2 is not loaded, in most programs, this loose item has no effect, although some programs that use the QMENU extensions can put the program to sleep in an unpredictable part of the screen.

In these cases, HITting the button allows you to move it as described above, DOing the button will reactivate the program again.

If the button is in the button frame, DO it to reactivate the program, HITting it will have no effect apart from redrawing the text written on the button.

5.3.3 SIZE

The SIZE loose item looks like this : 

Many programs do not have this loose item, their screen displays are fixed. If this item is present, there are a couple of ways that it can be used.

If you HIT or DO this loose item, the screen display will cycle through a pre-set range of suitable sizes and these are the only sizes that are allowed. This could be because the programmer wishes to retain some control over what size you make the display.

Other programs allow you to define the screen size as you see fit. By HITting or DOing this loose item, the pointer becomes the SIZE pointer and you can move it around the screen.

Moving the pointer down reduces the height of the display, up increases it, move to the right to reduce the width and to the left to increase the width. You can, of course, move down and right to reduce both the height and width. When you HIT or DO again, the screen will be redrawn to the required size - a small adjustment for character sizes and best fit may be done automatically.

A combination of MOVE and SIZE can allow you to make the display fill the entire screen area. (MOVE to the lower right corner then SIZE to the upper left one.)

5.3.4 WAKE

The WAKE loose item looks like this : 

This is another item that may not appear in many programs. If the program displays some information, but the information needs to be updated then HITting or DOing on this loose item will cause the display to be updated.

In addition, if this loose item is present and the program is reactivated (woken) from a button, most programs cause a WAKE to be carried out automatically.

An example of the use of this loose item is in a window that shows the files on a disc, perhaps. If you change discs nothing happens to the display until you re-read the directory. HITting or DOing this loose item will read the directory of the new disc and show its files in the window.

6. OUTLINE or PRIMARY WINDOW

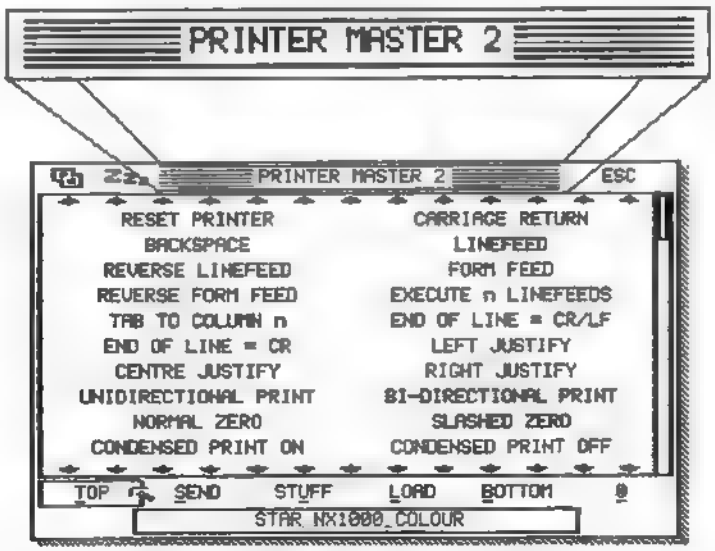
The **OUTLINE** or **PRIMARY WINDOW** of a program is quite simply the maximum piece of the entire QL's display that is used by the program in question. This is a bit of a simplification but covers it adequately.

Everything that the program does happens within the area of the screen defined as the program's primary window. All programs that have an OUTLINE set are deemed to be MANAGED and the PE takes care of them from there on. Most, if not all, PE programs handle this and the user need not be aware or worried by it at all. Basically, there is nothing that you need to do because you can't !

If you move the pointer outside of the current program's primary window, the pointer will change from the normal one to one of the others. Pressing a loose item selection key will now have no effect as you are no longer in communication with the program. Move the pointer back into the primary window and all will be well.

7. INFORMATION SUB WINDOWS

An **INFORMATION SUB WINDOW** is a small part of the program's display, which may or may not have a border around it, in which some information is displayed. The figure below shows a screen dump of a PE program showing an enlarged information sub window.



Actually, it is showing 2 information sub windows. The first is the large one with the striped (obligatory green and white stripes !) paper colour. There is nothing else in that particular sub window. The second information sub window is overlaid on the first. This one has the program name, Printer Master 2, as a text type **INFORMATION OBJECT** embedded in it.

An information sub window need not always have the same text displayed in it. The example above shows one which does, but at the bottom of the program's display there is another information sub window.

This one has a black border around it and the text 'STAR_NX1000_COLOUR' shown in it. As the program in question is used, this information sub window is used to show information particular to the function that the program is carrying out at the time. This is a bit difficult to show in a static document so you will have to take my word for it, or buy the program !

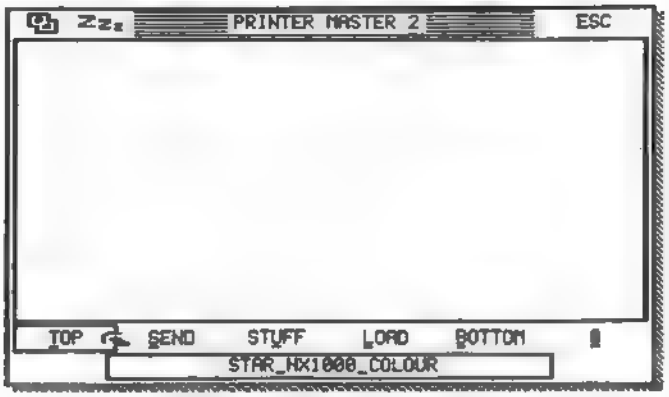
8. INFORMATION OBJECTS

As shown in the above screen shot, an information window can have some **INFORMATION OBJECTS** within them. These objects can be simple text or sprites (pictures - a pointer is just a sprite) The example above shows the text object 'PRINTER MASTER 2' in the enlarged information sub windows.

Information objects need not always be text. There are others as well, such as **SPRITES** or **PATTERNS** and **BLOBS**. These are discussed later.

9. APPLICATION WINDOWS

An **APPLICATION WINDOW** is simply an area of the program's display that is used by the program, the application, to do whatever the program does ! A drawing program, for example, will have a large area of its display set aside for drawing in. In the figure below, the application window is shown by the large shaded area.

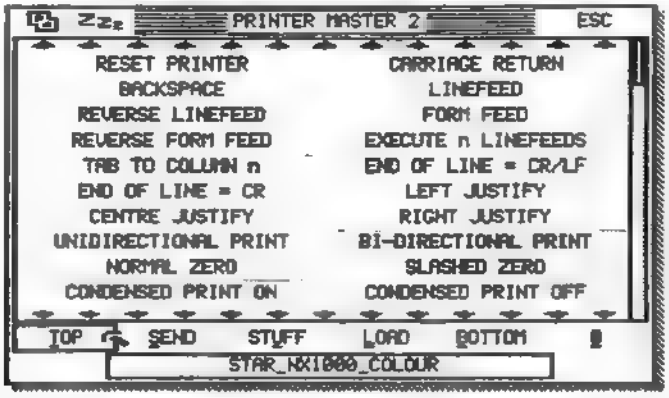


In many programs, an application window may be completely filled up by an **APPLICATION WINDOW MENU**.

10. APPLICATION WINDOW MENUS

An **APPLICATION WINDOW MENU** is just that. It is a menu, held entirely within an application window. Loose items, as discussed above, are sometimes referred to as loose menu items, but should not be confused by application window menus or application menus.

Once again, an application window is shown below in a screen dump from the program Printer Master 2.



As you can see by comparing this screen shot with the one above, the entire application window has been filled up with a menu. This menu is currently showing a number of **SELECTED** menu items. These are the ones that are shaded. The unshaded menu items are in a state known as **AVAILABLE**. Not shown on the above screen shot, is the **UNAVAILABLE** state for a menu item as this varies from program to program and from programmer to programmer.

An **AVAILABLE** menu item is one that the user of the program can select by HITting it. In many programs, DOing a menu item may also select it, but can be that DOing a menu item will cause some action to be carried out as well.

Of course, HITting a menu item can also carry out some action - it is all up to the programmer - read the manual !

Normally, a **SELECTED** menu item is highlighted in some way, in the above screen shot, the selected items are shaded to make them stand out. Selected menu items are the ones that are to be processed in some way at a later stage in the program. They may be a list of files to copy or delete etc. It all depends on what the program is doing.

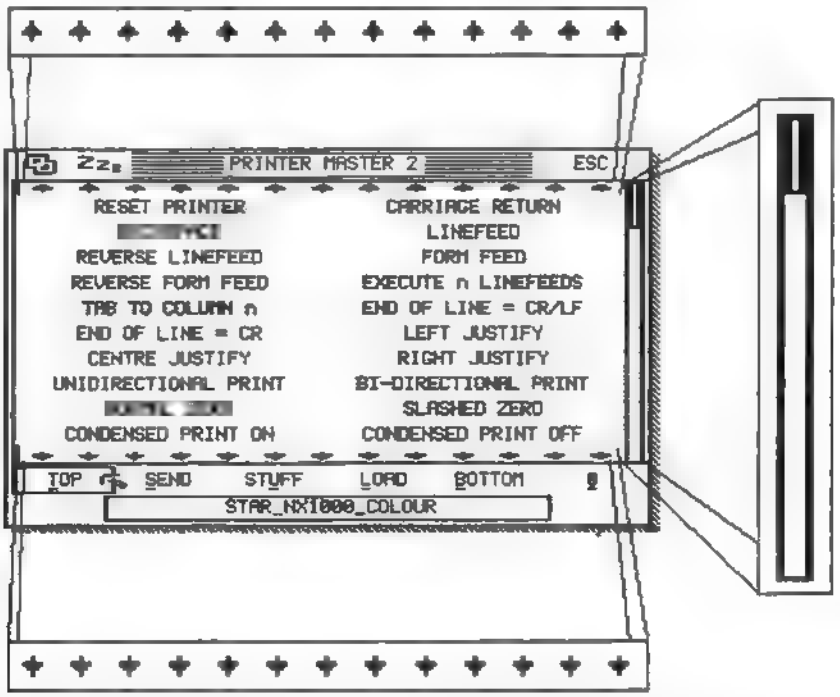
UNAVAILABLE menu items are the ones that are no longer able to be selected. If a program is deleting files from a disc, and has not updated its display, any file that has been deleted from the disc already (by the program) will not be able to be deleted again. These filenames will be made unavailable.

A lot of programs show unavailable items in a different INK and/or PAPER colour while others show the unavailable item by printing their names using an INK colour that is a stippled colour thus the file name, for example, would appear to be incompletely printed. HITting or DOing an unavailable item will have no effect.
(Loose items can also be available, selected or unavailable at any time - it is up to the programmer and usually depends upon what the program is doing at that time.)

When an application window is too small to show everything that is to be fitted into it, then SCROLL or PAN bars will be shown.

11. PAN and SCROLL BARS

In the screen shot above, there are a number of menu items showing in the application window menu. However, down the right hand side of the menu there is a position indicator while underneath and above the menu items there is a set of **SCROLL BARS**. The figure below highlights these.



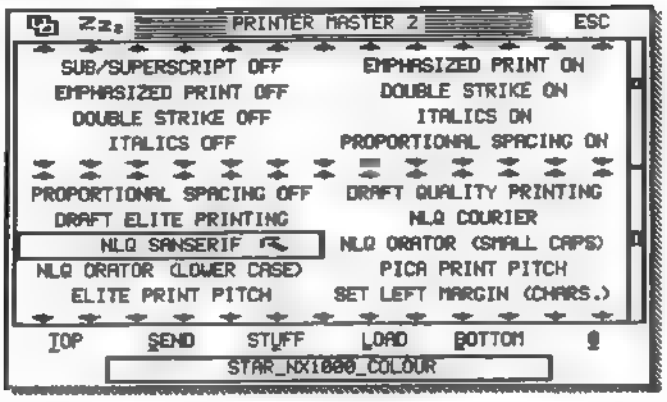
The scroll bars are used to move the visible area of the menu up and down so that the whole height of the menu can be accessed. By HITting on a scroll bar, the menu will scroll up or down by a single row. If you DO a scroll bar, the menu will scroll up or down by a whole page (minus one row). Instead of HITting and DOing, ALT and UP or ALT and DOWN scrolls one line in the appropriate direction and ALT SHIFT UP or ALT SHIFT DOWN pages in the appropriate direction.

Regardless of whether HIT or DO is used, the position indicator will show the position of the visible area of the menu relative to the height of the whole application menu. In our screen shot above, the visible area is at the top of the menu.

The position indicator can also be used to move around the menu. If you position the pointer over the scale area (above or below the actual indicator) and HIT it, the menu will scroll to that relative position.

If you move the pointer over the actual indicator and DRAG it up or down (DRAG means hold down the LEFT button or SPACE key and move the pointer while holding the button down), the menu will scroll up and down to follow the indicator.

Warning, HIT the position indicator, don't DO it. If you DO in the position indicator scale area, you may SPLIT the application menu. This is shown below.



As you can see, there appears to be two application menus in the same application window. If you look closely, there is a separator within the scale area which shows where the top half ends and the bottom half begins. Although each menu can be scrolled independently, they are both just separate parts of the same menu - but this allows you to see bits at the top and bits at the bottom all at the same time.

To rejoin the two halves, DO the separator.

You will notice that the size of the position indicator is relative to the size of the visible menu items relative to the total size of the entire menu. If the visible portion is a large fraction of the whole menu, the indicator is bigger.

Many programs do not allow an application menu to be split like this, others allow a number of splits.

PAN arrows and indicators are identical to SCROLL ones apart from their vertical orientation. PAN arrows allow the menu to be panned sideways as opposed to the SCROLL arrows vertical movements. The PAN indicator shows where you are relative to the width of the entire menu.

12. SPRITES, BLOBS & PATTERNS

These may be mentioned from time to time in PE program manuals. They are quite simple really, a **SPRITE** is a picture that appears on the display somewhere. A pointer is just a sprite that is moved around the screen. Sprites may be drawn to look like text, for example, in logos and programmer's names etc, or they may be small pictures to represent some function of the program - a picture is worth a thousand words etc.

BLOBS and **PATTERNS** seem to stick together a lot. They need to be combined in order to see something on the screen. A SPRITE can be thought of as a PATTERN combined with a BLOB to make a picture.

While a SPRITE has both shape and colour, the PATTERN has only colour information. The BLOB contains all the shape information.

While the SPRITE holds all its colour and shape data, a single BLOB can be used to supply the shape data for a number of shapes and by using different PATTERNS in combination, each combination can show the same shape in different colours.

If SPRITEs are used to display a row of squares, for example, each one will have the data for the square and the data for the colour. As the shape data is duplicated in each SPRITE it gets a bit wasteful if there are more than a few.

A BLOB that holds the shape data can be combined with a number of different PATTERNs and thus we have made a saving on the amount of memory required to show the row of squares.

13. PICKING PROGRAMS

Without the PE, you pick a required program by repetitive use of CTRL and C to activate the cursor in the next program in the list. Many times you will not know which program is now flashing and confusion reigns.

THE PE an idiot's guide - (CONT'D)

With the PE installed, picking a program is quite simple - move the pointer over some part of its display and HIT or DO it. Once this is done, you will see that the display for the chosen program is refreshed for you and you know exactly which program you are working in. Obviously, you can only pick programs that have windows open to the screen and that have an active cursor.

Beware, some programs, can set an OUTLINE that is much bigger than the menu that is displayed. If you move the pointer to a program showing under one of these, you will not be able to PICK it. What you are doing is PICKing the program with the large OUTLINE instead of the one that you want. These programs are slightly badly written !

14. FINALLY

That's about all there is to it. Once you have experience of using one or two different PE programs you should find that the rest are quite easy. As more and more programs are converted or written to use the PE, the QL is probably going to end up like the PC and become dominated by these types of program. Look on the bright side, however, compare the size of a QL program, even with the PE built in with one or two PC Windoze (!) programs - they now take up many megabytes (1024 Kilobytes) in order to be used.

My PC has a 340 megabyte hard disc that I have had to compress in order to get a few programs on to it along with Windows & Dos etc. My QL has a 40 megabyte hard disc and I still have not used much more that half of it. I have more useful programs on my QL than I have on the PC !

Professional & Graphical Software

NEWS

- There is now a Sbasic interface for PROforma
- LINEdesign now also has Qfax and LaserJet 4 driver

LINEdesign

This is the program which brought QL computing into the nineties. Finally you get access to a modern technology, "vector graphics." This means that your page will be stored in a mathematical form, and not as a collection of pixels
With LINEdesign, you can create artistic drawings, technical drawings, process bitmaps (even scale and rotate them!), and any kind of vector drawings. You can draw lines, curves, circles, ellipses, pies, squares, rectangles, rectangles with rounded corners, and any combination of these to create the most fabulous drawings ever seen. Because LINEdesign is a vector drawing program, any part of the picture can be moved, scaled, rotated, slanted without any loss of precision or resolution. In LINEdesign, pictures are device independant, meaning that the printout will be the same on any printer (e.g. same size and position).
Also LINEdesign is good at handling text. You can easily put titles and full paragraphs on the page. You can choose from a large variety of fonts (you get 130 with the program), and they can be displayed at any size, rotation, etc. If the fonts which are given with the program are not enough for you, there is a special program to convert Adobe Type 1 fonts for use by LINEdesign (pfb2pff).
LINEdesign is a drawing program, but it can also be used by people who are not good at drawing. LINEdesign is a great program for making leaflets, posters, and any kind of printed work. To add a graphical touch, you get about 150 clipart pictures, including banners, borders and general purpose drawings. LINEdesign will reproduce everything at the highest possible quality !
LINEdesign is delivered with an extensive manual, which includes a full printout of all the fonts and the clipart given with the program.

PROforma

PROforma is a vector graphics library for C (and Assembler) programmers. It is very powerful, and can be used for any application which needs high quality output. PROforma is the graphics library which is used by LINEdesign, PFdata and PFlist to produce the output. PROforma supports black and white vector graphics and includes :

- clipping paths
- transformation matrixes
- thick lines, grayshades, lines and bezier curves
- filling using even odd and winding rule
- vector (outline) fonts, which can be used in any size. Hinting is used to make sure small fonts look good.
- true WYSIWIG. PROforma can generate output for screen and printer, and the output will be exactly the same on both (with any difference due to difference in resolution)
- bitmaps. Although PROforma is a vector graphics library, you can include classic bitmaps so you can still use your old graphics.

The PROforma package contains a dedicated C68 library to access the PROforma extension thing. It is supplied with a comprehensive manual and examples.

- PFlist : Very easy to use program to create listings on any printer (especially inkjet and laser printers). Can include a footer with filename and filedate. Always allows perforation of your pages. The font and fontsize can be chosen (PFlist uses PROforma). PFlist can create your listings in two columns, and in landscape (or both).
- DATAdesign : This is probably the most userfriendly database program in the world ! Never before has it been so easy to create, fill in and maintain your personal databases. If you want to start a new file, just type the names of the fields. If you want to add or delete a field, no problem, just do it. If you want to change the name of a field, just indicate it.

What's more you can choose to look at only those fields you want, and in any order you specify. And you can select which records you want to view, and which not.
DATAdesign allows you to have some hidden comments for each record, have a general look at the file (in tabulated form) or to transfer a record into the scrap or hotkey buffer, so you can easily import a record in your favorite word processor or editor !
Security is another strong point for DATAdesign. Usually files will be memory based, which offers you maximum speed. However, files can also be disk based, making sure that all changes to the file are immediatly stored on disk, so even in the event of a power failure, you can at most lose the changes to one record !
Naturally, DATAdesign is good at sorting and searching. And if you were using another database, you can convert Archive or Flashback files to DATAdesign.
Wait no longer, start using DATAdesign !!

- DATAdesign API : Unleash the real power of the DATAdesign engine. For programmers, DATAdesign turns into a relational database with a bonus, you don't even need a key field. The Application Programming Interface allows you to use database capabilities without learning a new language, you can just use the extensions to SuperBasic, C or Assembler. The API gives you a unique and powerful record at a time data manipulation language !
- PFdata : Very interesting program for all DATAdesign owners. This program can create hardcopy of your DATAdesign files using PROforma. This means that you can use a large selection of fonts (such as the ones included in LINEdesign), in any requested size. Also LINEdesign pictures can be included to add logo's, boxes, etc. Several records can be printed on each page,...
- prices : PROforma fontpack BEF 4000, PFdata BEF 1000, PFlist BEF 1000, pfb2pff BEF 3000, LINEdesign BEF 5000, PROforma BEF 5000, DATAdesign BEF 3000, DATAdesign API BEF 1000, postage and VAT included. Outside EEC : call! Send Eurocheque in BEF, or your VISA/ EuroCard/ MasterCard details.

for more information, just write, phone or fax !!

Haachtstraat 92, 3020 Veltem, Belgium, tel/fax (016) 48 89 52

THE QL COLLECTION

	£
3D PRECISION COMPUTER AIDED 2D/3D DESIGN SYSTEM	49.95
ADVENTURE CREATION TOOL SPECIAL EDITION SYSTEM INCLUDING IMAGINE ADVENTURE	49.95
ARCADIA GAME WITH BMX-BURNER & GRIDRACER	9.95
ARCHIVE DEVELOPMENT SYSTEM + RUN-TIME MODULE	29.95
ARCHIVE TUTORIAL SYSTEM FOR BEGINNERS TO EXPERTS	19.95
BIBLE EXTRACTS 1.5 MB TOTAL (KING JAMES VERSION)	9.95
BETTER BASIC EXPERT SYSTEM AUTO PROGRAM OPTIMISER	24.95
BLOCKLANDS GAME WITH 65,536 4K x 8K SCREENS	9.95
CASH TRADER v3.3 + ANALYSER (QL TRADING ACCOUNTS)	99.95
COMPARE AUTO/MANUAL FILE MANAGEMENT UTILITY	19.95
COPY UTILITY AND PRODUCTIVITY ASSISTANT	9.95
CPORT SUPERBASIC TO C AUTO TRANSLATION SYSTEM WITH CFIX IMPROVER	89.95
DATABASE ANALYSER FOR ARCHIVE & STARTREK INFOBASE	24.95
DAT-APPOINT APPOINTMENT ARCHIVE DATABASE SYSTEM	19.95
DICTIONARIES FOR PERFECTION PLUS SPECIAL EDITION WITH PRIME DISTRIBUTION PROGRAM	9.95
DIGITAL C SPECIAL EDITION COMPILER SYSTEM WITH ENHANCED MACHINE CODE LIBRARIES	49.95
DISKTOOL WITH QUICKDISK DISK UTILITY	19.95
DROIDZONE GAME WITH ACCELERATED GRAPHIC ENGINE	9.95
EDITOR SPECIAL EDITION ALL-FILE (INCLUDING NON-ASCII) TEXT PROCESSOR	49.95
EYE-Q STATE OF THE ART QL GRAPHICS SYSTEM	39.95
FONT ENLARGER ACCESSORY FOR PRO PUBLISHER	19.95
GRAFIX QL 24-PIN UNIVERSAL PRINTER DRIVER	14.95
HARDBACK + FINDER HARD DISK BACKUP ENGINE	49.95
HEXDUMP UTILITY FOR VIEWING AND ANALYSING MEMORY CONTENTS	9.95
HP PRINTER DRIVER FOR PERFECTION PLUS SPECIAL EDITION	9.95
IDIS INTELLIGENT DISASSEMBLER SPECIAL EDITION	39.95
LIGHTNING GOLD SPECIAL EDITION SUPER-ACCELERATOR SYSTEM	39.95
MAILMERGE DATABASE SYSTEM FOR ARCHIVE	19.95
MEDIA MANAGER SPECIAL EDITION EMERGENCY & MANAGER KIT	49.95
MENU EXTRACTION PROGRAM FOR COMPRESSED DISKETTES	9.95
MICROBRIDGE CONTRACT BRIDGE BIDDER & PLAYER	39.95
NAMES AND ADDRESSES ARCHIVE DATABASE SYSTEM	19.95
PAYROLL SYSTEM FOR UK SMALL COMPANY PAYE	49.95
PC CONQUEROR GOLD SPECIAL EDITION (1.5MB RAM SYSTEMS)	99.95
PC CONQUEROR STANDARD VERSION INCLUDING XOVER MULTIMEDIA CONVERTER	59.95
PEDIT PRINTER DRIVER FOR PSION XCHANGE SUITE	19.95
PERFECTION PLUS SPECIAL EDITION WORD PROCESSOR WITH SPELLCHECKER	139.95
PERFECT POINTER TOOLS ACCESSORY FOR QRAM/QPTR USERS	29.95
PROFESSIONAL ASTROLOGER PREDICTION & DELINEATION SYSTEM	59.95
PROFESSIONAL ASTRONOMER INTEGRATED SYSTEM	29.95
PROFESSIONAL PUBLISHER DESKTOP PUBLISHING SYSTEM	89.95
PROFESSIONAL PUBLISHER TOOLBOX PART ONE	29.95
PROFESSIONAL PUBLISHER TOOLBOX PART TWO	29.95
PROFESSIONAL PUBLISHER TOOLBOX PART THREE & CLIPART	29.95
QFLICK CARD INDEX QPTR DATABASE SYSTEM	29.95
QKICK FRONT-END SYSTEM & DESKTOP MANAGER	24.95
QL DATA FILE COMPENDIUM FOR USE WITH DEMO PERFECTION	19.95
QMATHS MATHEMATICAL SYSTEM PART ONE	69.95
QMATHS MATHEMATICAL SYSTEM PART TWO & COMPENDIUM	59.95
QMON QJUMP MACHINE CODE SUPER MONITOR	39.95
QUICKLASER PUBLISHER DRIVER FOR HP LASERJETS & DESKJETS	19.95
RECOVER ENGINE FOR CORRUPTED ARCHIVE DATABASES	19.95
REVERSI / OTHELLO GAME - WORLD CHAMPION STANDARD	9.95
SEDIT + SCREENPRINT ARCHIVE DEVELOPER UTILITIES	19.95
SUCCESS CP/M & HIGH SPEED Z80 EMULATION SYSTEM	49.95
SUPERBASIC MONITOR AUTO RUN-TIME TRACER/ANALYSER	24.95
SUPERFORTH FORTH-83 COMPLETE COMPILER SYSTEM	39.95
SUPER BACKGAMMON GAME - CLUB PLAYER STANDARD	9.95
SUPER SPRITE GENERATOR GRAPHIC TOOLKIT	29.95
TOOLKIT III SUPPLEMENT TO SUPERTOOLKIT II COMMANDS	29.95
TRANSFER UTILITY SPECIAL EDITION WITH MODIFIER	29.95
TURBO SUPERBASIC COMPILER v3.24 + TURBO TOOLKIT	99.95
ULTRAPRINT SCREEN DUMP ACCESSORY FOR EYE-Q	19.95
UNZIP v5.0.1 PD FILE DECOMPRESSION / RETRIEVAL UTILITY	0.00
XREF SUPERBASIC AUTO PROGRAM ANALYSER & REPORTER	29.95
ZIP v2.0.1 PD FILE COMPRESSION & ARCHIVAL UTILITY	0.00
TOTAL OF THE ABOVE PRICES	£2,321.80

ALL THE ABOVE FOR JUST £179!

WHAT WILL THE QL COLLECTION COST ME?

Just **£179 in total**. There is nothing to add, no hidden taxes, and P&P to anywhere on earth is included (add £5 for Airmail). You save over £2,100. DP recognises that you probably have some titles already (though perhaps not the latest releases), and some may not be of interest to you yet (likely to change when you see them!) - the price reflects this! **THE QL COLLECTION** is worth it even if you only want to update all your existing DP products: however, DP will continue to accept orders for *individual* DP programs at the prices quoted if you really insist.

WHAT EXACTLY IS INCLUDED IN THE QL COLLECTION?

You get the fullest, very latest, most up-to-date releases of all - **every single one** - of the **66** QL programs listed. The software is the finest, and the QL's very best. The titles would cost you over £2,300 (plus applicable P&P) to buy individually - you can check this by summing the prices overleaf, or those quoted in earlier ads. The only titles omitted are Mega Dictionary (only for 2Mb RAM systems - add £15 for it), MS-DOS v6.22 upgrade (add £90) and any less capable variant of a title that is itself included in **THE QL COLLECTION** (e.g., since the top-of-the-range PROFESSIONAL PUBLISHER is included, DESKTOP PUBLISHER is obviously excluded). You get both versions of PC CONQUEROR (as the list shows) so as to cater for all hardware variations. You may never ever need to buy another QL program again. The range of software you will get is truly staggering. It is too good to be true. But it is true - while the offer lasts....

WHAT ABOUT THE PROGRAM DOCUMENTATION?

All the latest applicable documentation (lots and lots of it) is included on disk, and can be read and printed using Perfection Special Edition or Editor Special Edition, which are also both included, and which can - of course - be used to search, browse, analyse or "edit" manuals at your leisure. Printed copies may be bought later if wanted - full details are sent with the order.

WHY CAN'T I FIND THE CATCH IN ALL OF THIS?

Because there isn't any. DP, whose QL commitment continues, makes this super offer to celebrate the birth of a marvellous baby daughter Michelle, now through all her early problems, to Julie and Freddy. **THE QL COLLECTION** is licensed for use by the purchaser alone, who by buying it agrees not to resell or otherwise pass on any part of it, or of any DP software already possessed. Technical support is negotiable: full details are supplied with the order for you to take up should you want to do so. DP *reserves the right* to withdraw **THE QL COLLECTION** offer at any time later than 14 days after your receiving this magazine, so please do hurry.

WHAT HARDWARE WILL I NEED TO RUN THINGS?

You will need a twin disk drive (DD, HD or ED, 3.5" or 5.25"), lots (123?) of blank disks and over 1.5Mb RAM (Gold Card, Super Gold Card, QXL, ST/QL and equivalents) to fully use **all** the software. The vast majority of titles will, however, work on much smaller systems: earlier DP ads indicate with precision the minimum hardware needed to run each program. If no disk size is specified when ordering, DP will assume 3.5" DD. If you do not yet have a powerful enough QL system, you may wish to contact a hardware dealer and buy, say, a second-hand Gold Card and/or twin disk drive (preferably 3.5" DD or HD - avoid Mitsubishi, and if HD or ED, ensure 100% compatibility with all Gold Cards is *fully guaranteed* by the supplier) as needed.

HOW CAN I GET MY COPY OF THE QL COLLECTION?

THE QL COLLECTION can only be obtained directly, by posting your order (including payment of £179 by cheque drawn on a UK bank / building society, Eurocheque or postal order, or quoting a VISA or MASTERCARD credit card no: and card expiry date) as soon as possible to:

DIGITAL PRECISION LTD, 222 THE AVENUE, LONDON E4 9SE

THE QL COLLECTION

Town Crier Announcements of Upcoming Events

To have your Show, Workshop or AGM listed by the Town Crier, send all relevant information to IQLR's North American address. Please note deadline dates for submissions listed on page two of this issue.

4 March 1995 ***(SUNDAY)***

INTERNATIONAL QL SHOW

(Sponsored by: Sin_QLAir)

St Joris College
Roostenlaan
Eindhoven
NETHERLANDS

26 March 1995 ***(SUNDAY)***

QL WORKSHOP/SHOW

Contact: Phil Jones
Tel: 0602 730142

Long Eaton
nr Nottingham
GREAT BRITAIN

30 April 1995 ***(SUNDAY)***

QUANTA AGM

Walton Park Hotel
Clevedon
Bristol
GREAT BRITAIN

7 May 1995 ***(SUNDAY)***

6th ITALIAN QL SHOW

Contact: Davide Santachiara
Tel: +39 522 300409

Reggio Emilia
ITALY

10 June 1995 ***(SATURDAY)***

3rd NORTH AMERICAN QL SHOW

Contact: Bob Dyl
Tel: +1 401 849 3805

On Site Contact: Mel LaVerne
Tel: +1 615 483 4153

Faith Luthern Church
1300 Oak Ridge Turnpike
Oak Ridge, Tennessee
USA

OLIMPO: multiplatform development in C

Madrid, SPAIN - Pedro Reina

There are different kinds of computer architectures, operating systems and personal configurations, but all of them are intended to do the same thing: running your favorite software. When you find (or write) a program that is worth to you, you would like it to run on any computer, not just your QL, Mac or whatever you have.

The biggest software companies usually offer their products for different operating systems. You can find "that" program on OS/2, UNIX, System 7, and it works almost the same on all of them.

But, how can they, and you, write a program for different computers? You can of course write different versions for each computer, but this leads you to big trouble with code maintenance and revision. A far better solution is to write only one source version and compile it with different compilers for each operating system required.

There are many commercial tools that allow you to write code once and compile it on many computers. For example, just browsing the last issue of "Dr. Dobb's Journal" I find these products:

* zApp, from Inmark Development Corporation, for Windows, OS/2, UNIX/Motif and DOS.

* XVT, from XVT Software Inc., for Windows, OS/2, Macintosh, UNIX and DOS.

* object-Menu, from Lifeboat Publishing, for Windows, OS/2, and DOS.

* StarView, from StarDivision, for Windows, Macintosh, OS/2, and UNIX.

* MEWEL, from Magma System, for Windows, OS/2, UNIX, VMS and DOS.

There are many more, but I get bored when browsing. As you can see, there is no one that could help you to build a QDOS executable.

In this article, I will explain how you can write your own development system which you can use to write once your program and compile it on a QL and a PC. I will also show what is my personal system, called "Olimpo", and what kind of help you can expect from it.

If you are an expert C programmer, you will be able to implement your own system for other computers, not just the QL and the PC; if you are a novice with C you can start just now to use Olimpo and improve your programming productivity.

Writing portable code - First of all, we need a language as portable as posible. We can't expect that a SuperBASIC program "as is" will run on a different interpreter.

The C language is a very good programming language: it is powerful and fast, there are a lot of tools for it and you can find a C compiler on almost every computer. It was designed to be non machine dependent. If you follow strictly the C standard, it is very likely that your code will compile on different computers, operating systems and compilers.

Of course you can write non-portable code, if you make wrong assumptions. For instance you can assume that an integer value will take 2 bytes, but this is not true, it can take 4 bytes as this is up to the individual compiler. You must know these kind of things and don't use compiler-specifics sizes, alignements, evaluation order and the like. There is a lot of literature about writing good standard C and much more good public domain code you can study.

Anyway, you should not have much trouble with your first programs in C; you can code clean from the start, as C is not an "obscure" language. In the more simple cases, translating a SuperBASIC program structure to a C one is straigthforward. You can see at Listing 1 a program that print on the screen the odd integers from 3 to 15. As you can see, they are very similar. This is a toy example, but it is sure that you can write hundreds of lines like that in C and compile them without change on many computers.

OLIMPO - (CONT'D)

Listing 1 SuperBASIC version

```
10 FOR i = 3 TO 15 STEP 2
20   PRINT i
30 END FOR i
```

```
C version
```

```
int i;
for ( i=3 ; i<=15 ; i+=2 )
  ( printf ( "\n%d", i ); )
```

Programming for the QL and the PC - Following just the C standard is good, but it is also very restrictive. There are a lot of things that simply you can't do as there is no standard C function that meets with them. For instance, you can't put the cursor everywhere you want on the screen, nor generate different sounds (just one, with printf("\a")) nor wait for the user to type the 'F1' key.

However, the modern compilers do have many functions you can use when stardard C is not enough. Usually this functions have different names and behaviour from one compiler to another. So, how can we use these specific functions and still writing portable code? The solution is using conditional compilation. The C language has a powerful feature, the preprocessor, witch we can use to strip out at compiling time the code non related with our target computer and stay on the code the statements required on our computer.

This is a key explanation of this article. At the top of your code (or with the -D option) you define with a macro the computer you are compiling on:

```
#define QL   /* Use QL or PC */
```

when you need to insert QL or PC specific code, you use code like this:

```
#ifdef QL
  /* QL specific code */
#endif

#ifdef PC
  /* PC specific code */
#endif
```

With this scheme, we can start to use platform dependent code without any problem. Now we can take advantage of the goodies that our compilers offer for us to extend the standard functions.

From now on, I will focus on QL and PC code writen with the compilers I use. If you have different compilers, you should make little modifications to my code.

On the QL I use C68, an excelent public domain compiler that you can obtain from most PD software suppliers. I must thank to Dave Walker and the C68 team for his excelent work and kind support. On the PC I use a cheap compiler from Borland, Turbo C; currently I use Turbo C++ 1.01.

How can we have our own funtion to locate the cursor on the screen? Sure, you have a function in C68 that do the job and another one in Turbo C. All you have to do is choose your own name for the function as you wish. Let's choose SetCursorAt().

At last, the actual code is:

```
#ifdef QL
#define SetCursorAt(r,c) sd_pos(fgetchid(stdout),-1L,(int)r,(int)c)
#endif
```

```
#ifdef PC
#define SetCursorAt(r,c)  gotoxy((c)+1,(r)+1)
#endif
```

when 'r' stands for "row" and 'c' for "column". Note that upper left corner is 0,0 in C68 but 1,1 in Turbo C. The function SetCursorAt() will use the C68 convention; this explains the "+1" of the PC version.

Now you can use the new function in the rest of your code: we have so isolated a specific feature and we have created the first function of our own system.

This technic works fine for both simple functions and constants. Do you want to use the green color for something? Define the constant macro GREEN using this code:

```
#ifdef QL
#define GREEN  4
#endif

#ifdef PC
#define GREEN  2
#endif
```

Not just you can paint with green color on both computers, you also have a more readable code writing "GREEN" rather than "4" or "2".

There are a lot of situations where a macro is not smart enough. You must use a real function. No problem: use conditional compilation and keep your function on a separate source file. Compile it alone and then link the object code with your main program.

For instance, you need a funtion called DefineScreen() that prepare the main screen for your program. Probably you want to use the same screen dimensions on both computers. Write a file called, say, DefScr_c (use at most 8 characters apart the extension in orden to keep the same names on QL and PC). You can see on listing 2 an example of such a file. Compile it using the -c option of your compiler and you get a file called DefScr_o (or DefScr_obj on a PC). When you compile your main program, add the module DefScr_o. If you need to use many specific functions (and you will need it) you should group together all your object modules building up a library file.

Listing 2

```
/* Define a screen of
 * 25 rows and 80 columns
 */

#ifdef QL

#include <qlib_h>

DefineScreen()
  {
  struct QLRECT Block;

  Block.q_width  = 480;
  Block.q_height = 250;
  Block.q_x      = 15;
  Block.q_y      = 3;

  sd_wdef (fgetchid(stdout),
           -1L,0,0,&Block);
```

OLIMPO - (CONT'D)

```
  return ( 0 );
  }

#endif


#ifdef PC

#include <conio.h>

DefineScreen()
  {
  union REGS r;

  r.h.ah = 0;
  r.h.al = 0x03;
  int86 (0x10,&r,&r);

  return ( 0 );
  }

#endif
```

At last, you will have your own system with:

1. A header file with your function-like macros, your constant macros and the declarations of your real functions.

2. A library file with the object code of all your functions.

And for you to use the system just have to include in your programs the header file and link with the library file. The differences between your computers have been isolated, so you can write very clean code using your own names for all your functions. It's easy! It works! Try it!

The object-oriented approach - The object-oriented programming (OOP) is a well fashioned programming paradigm. There are object-oriented languages, as C++, Ada, Eiffel and SmallTalk, but they are not required to do OOP; they of course help the programmer to coding using the paradigm but you can also use the very good ideas the OOP gives you by means of other programming languages.

Encapsulation and inheritance are two of the major aspects of the OOP. Encapsulation means that you group together the data and the methods that apply to these data. You hide the implementations details in your code and declare the interface of the methods you can call from the outside. You so built an "object": the internals of it are up its programmer; the code that uses an object just knows about the public interface of it. Using inheritance you can derive from a basic object (a "class") a more specific object, inheriting some of its methods, modifing some others and adding new ones. I always use encapsulation. There are some objects that are obvious when programming: for instance, the screen. There are many functions related with the screen, but when you have managed to write them, you no longer need to know how they work to use them. The screen is an object and it is worth encapsulate all its internal programming and just say how you can use it.

This approach helps you very much when you are intended with multiplatform programming. The screen object is absolutly hardware-dependent, so it is worth to hide the details and use the abstraction: you have a screen, you can do a lot of things with it just using the correct function, but how they are really done is not up to you, the programmer of the object knows (that programmer may be either yourself, or a friend or even a commercial software supplier) and, this is important, he or she knows how to do the work by different way on different computer, but getting the same result.

QLIMPO - (CONT'D)

I think that it is very helpful starting all the functions (methods) of an object with a brief indicator of that object. For instance, we can short "screen" with "Scr" and re-call our two above defined functions like that: the SetCursorAt() becomes Scr_SetCursorAt() and ScreenDefine() is now Scr_Define().

I have been working for more than a year following that guidelines, writing code for QL and PC, but with an eye on UNIX, always thinking about porting my code to COHERENT, a very cheap UNIX clone from Mark Williams Company. I have found several hardware-dependent objects:

Screen. It covers all matters related to printing on the screen, using of color and highligting, cursor positioning and enabling, partial and full cleaning.

Time. Reading of system clock, precision timing references.

Sound. It knows how to generate a number of different sounds.

Character. It handles the differences between the character tables on QL and PC, and extends the C functions related with characters but witch only work with characters up to the 127.

Key. It hides all the differences between a QL and a PC keyboard. The main feature is that the programmer can use a function called Key_Pressed() and it just returns a number that you can compare with the macro you want: KEY_F1, for instance.

There are more issues that must be isolated, but I think that with these basic objets you get the idea. It's easy to start implementing these objects and then build up a good application absolutly portable across QL and PC.

Further work - If you are reading so far, I'm sure that you are interested on building good apps. So do I. Fine, as we have arrived to a crucial point. Now we can start to use the basic objects we have just developped for implementing more complex objects. Our further work would be hardware independent, as it is already based on our abstractions. A clear instance is the Menu object. Think about: what do you need to write a function that implements a menu? Let's see:

You need to write options on the screen, probably with different color for the selectioned option and highliting the mnemonics. Perfect, we have the Screen object which meets with all these necessities.

You need to process the keys pressed by the user, allowing him or her to navigate through the options with the cursor keys or selecting directly an option by pressing its mnemonic key. An easy work for the Key object.

You need to know the upper and lower case of the hotkeys, to accept any case as valid input from the user. You can use the Character object for that.

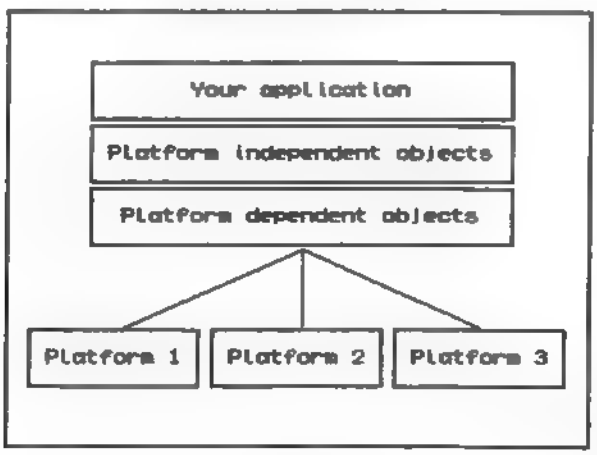


Figure 1

It would be nice if you generate a little acoustic warning when the user press a not allowed key. You do know how to solve this.

The same way you can build the Menu object, you can build many other interesting objects and add them up to your system. If you are careful, you can design generic objects that you can then use over and over again. All of them will be portable to any other platform you want, as long as you first adapt the basic objects to the new platform.

Please refer to figure 1: you are at the highest level building your app. You can use platform dependent and independent objects, but at the low level only the platform dependent objects communicate with each platform.

OLIMPO - (CONT'D)

Olimpo: hands at work - All the explained above are what you could do, but these ideas have already been developped. As you know, I have named my system "Olimpo". It has 17 objects, summing 156 functions, all of them working on QL and PC. The PC can be used in text mode or in graphics mode (VGA required).

All the source code, screen messages and documentation of Olimpo are in Spanish, but I think that it could be useful if you really need QL and PC source code compatibility.

This is a brief description of the Olimpo's objects:

Screen, Time, Sound, Character and ***Key*** as explained above.

Random. Generation of random numbers.

List. Linked lists handling.

Zone. I call "zone" a region of the screen when you can draw graphics. You can light every pixel of a zone on a QL screen or on a VGA graphics card on a PC. I'm writing code to draw resolution-independent shapes.

String. As expected, this is a collection of functions string related.

Box. You can use this object to draw boxes and lines on the screen and to generate boxes to be printed.

User. This is a collection of functions that you can use to interact with the user of your program. For instance, you can inform an error, ask a yes/no cuestion, edit a string, and the like.

File. This object copes with the differences between the file handing on QL and PC. The object also traps the errors can occur when you open, read or write a file.

Menu. You can ask for a horizontal or vertical menu and this object just returns to you the option choosen for the user, hidding all the details.

Program. This object is very simple. It just presents your program to the user: name, version number, author and release date are presented on screen.

Configuration. Sometimes is good to have a configuration file that your program can read in order to change its behaviour. I thought that it was desirable that the configuration files could have embedded comments. This object reads an ASCII file stripping out all the comments, so the programmer just has to handle the real data.

Database. Databases are needed very often. There are a lot of formats in the computer world for a database. I have choosen a widely accepted format: the so called xBase-compatible. Olimpo can create, read and modify database files compatible with dBase III, III+, IV and ***Clipper***. That means that your data could be read and modified with a lot of commercial tools. And you can read and modify with Olimpo external data witch use this format.

Index. xBase compatible files are not indexed, so an external index file is required to mantain an order on the data. There are different formats for these index files. Olimpo can read NTX files, the format used for the Clipper language. I'm also writing code for creating and modifying NTX files.

Using Olimpo - It's time to see a little example of the real code you can write with Olimpo. I have translated to English all the function names and screen messages specialy for this occasion. Please take a look to listing 3; remember that each function starts with three letters identifing the object. I hope that the code was clear, even if you are not a C programmer. If you can't understand almost everything of this little program, I have not succeeded implementing Olimpo.

Listing 3

```
#include <Olimpo.h>
```

OLIMPO - (CONT'D)

```
main()
  {
  char * Message;
  int    Lenght;

  Scr_Define ( SCR_TEXT );
  Prg_Present ( "IQLR example", "1.0",
                "Pedro Reina", "1994" );
  Box_Color (WHITE);
  Box_Frame (BOX_SIMPLE,1,0,21,79);

  Message = " Example for International QL Report ";
  Lenght  = Str_Lenght (Message);
  Box_Frame (BOX_DOUBLE,3,20,7,25+Lenght);
  Scr_Color (WHITE,BLACK);
  Scr_SetCursorAt (5,23);
  Scr_Text ( Message );

  Box_Color (GREEN);
  Box_Draw (BOX_DOUBLE,BOX_SIMPLE,
            2,2,3,8,12,25);
  Scr_Color (BLACK,WHITE);
  Scr_PutText (10,32,"System clock");

  Scr_Ink (GREEN);
  Scr_PutText (13,28,"Year");
  Scr_PutText (13,36,"Month");
  Scr_PutText (13,46,"Day");
  Scr_PutText (16,28,"Hour");
  Scr_PutText (16,36,"Minute");
  Scr_PutText (16,45,"Second");

  Scr_Ink (WHITE);
  Scr_PutInteger (14,28,Tim_Year(),4);
  Scr_PutInteger (14,37,Tim_Month(),2);
  Scr_PutInteger (14,47,Tim_Day(),2);
  Scr_PutInteger (17,29,Tim_Hour(),2);
  Scr_PutInteger (17,38,Tim_Minute(),2);
  Scr_PutInteger (17,47,Tim_Second(),2);

  Usr_PressAnyKey ( "Program finished" );
  Scr_Close ();
  return ( 0 );
  }
```

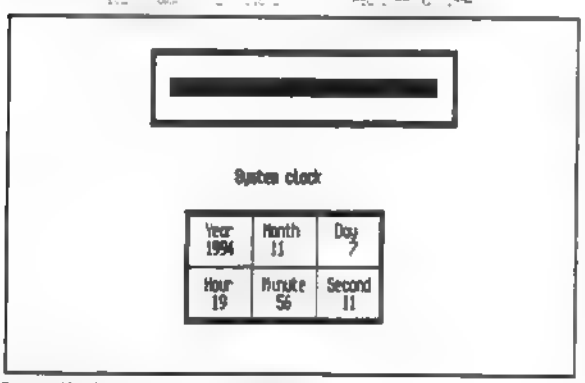


Figure 2

You can see at Figure 2 the output of the program on a QL screen. You can now transfer the code to a PC, recompile and... presto! You have the same program running on it.

In order to help Olimpo users to get started, the Olimpo distribution discs contain a long demo source file with some examples on how to use the different objects of Olimpo.

You can see at Figure 3 a screen snapshot of the program when it is showing an example of the Index object. At the

OLIMPO - (CONT'D)

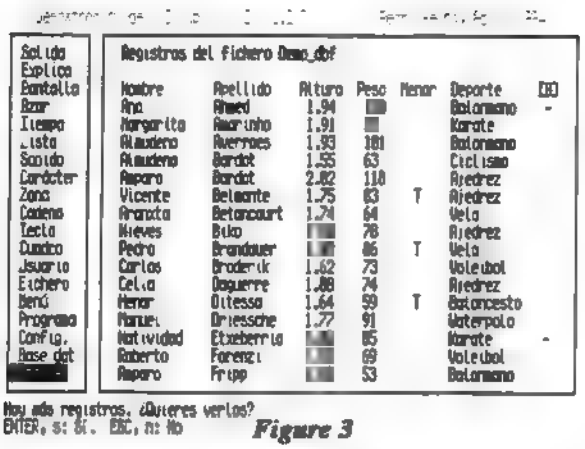


Figure 3

top, the function Prg_Present() shows the name of the program and related data. At the left, the main menu with the "Index" option highlighted. At the right, some registers of a database file are printed on the screen following the order read on an NTX file. At the bottom, the function Usr_Consult() is asking something to the user.

I have been using Olimpo for more than a year, building up a dozen of programs for personal and commercial use. I feel very comfortable with it, as it was designed following my personal taste. But the most important thing is that I can develop my apps much faster, as I have many complex functions always available.

When I'm writing a program with Olimpo, I'm free to use any of my computers. Often, I start the program on one computer and finish it on another one. My main computer is a Spanish ROMed QL with Gold Card and ED drives. I have a 66-MHz PC that I use when I need to recompile a program a lot of times or for debugging. When I go to holiday, I use a 386 notebook. Olimpo lets me transfer freely code anywhere I want.

There are very few people using Olimpo at present. I hope that the number will increase in the future. I have found that Olimpo is useful on very different apps. You can see a few shareware programs that use Olimpo at the QLIPER Disk (Special 25 of the QUBBESoft P/D Catalogue). Salvador Merino (from Spain) has used Olimpo on a few of his interesting projects, as a graphics database called "Foto-dBase" and a threaded interpreter called "MERINO TIL".

Conclusions - We can use our QDOS systems for writing programs which run on other operating systems. The commercial and expensive multiplatform tools don't cope for QDOS, but we can make our own multiplatform system using the C language and the C68 compiler. It will fit our taste as it will be developped from scratch following personal preferences.

But such a system already is running; it is free and can be adapted or used as reference. There are an increasing number of programs with source code portable between QL and PC written with it.

Using a set of primitive functions that isolate the platform specific features of the different C compilers, we can build a powerful development system. We can increase our programming productivity using well designed and reusable pieces of code. An object oriented approach could increase even further the quality of our code.

At last, it is possible to write better code, more reliable, absolutly portable, spending far less time just following simple guidelines.

How to get Olimpo - I have put Olimpo on the Public Domain, hoping to help the QL programmers that could be interested.

There are two discs available: in QL and in PC format. Both contain full documentation, and include library files different for each computer (you need C68 on the QL or Turbo C++ on the PC), the demo files and full source code in compressed form.

Olimpo can be obtained from QUBBESoft P/D (thank you, Ron!). With the two main discs you get an "Examples Disk" witch contains source code and executables of some programs I have written with different versions of Olimpo.

(Editor's Note: Pedro was kind enough to send IQLR a complete set of the OLIMPO disks (which we will add to the library). Would anyone like a chance at translating the doc's and screen prompts into English??)

AMADEUS INTERLINK

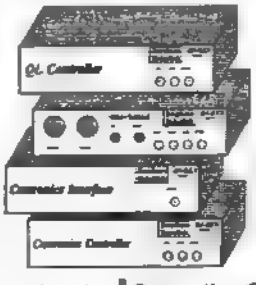
has arrived!

The ultimate expansion for the QL

Now *easily* connect up to 255 I/O interfaces to the QL. The Amadeus Interlink system is a local area network capable of linking up to 255 devices such as computers *(PC's & QL's at the moment, others to follow)*, Centronics interfaces (bi-directional parallel printer ports), Sound interfaces, RS232 ports and other useful I/O interfaces.

Gone is the jungle of QL expansion problems! Amadeus system interfaces are housed in small, smart, black enclosures that are easily stacked on their sides or tops. Approx. measurements 4.3"x2.3"x1.2" (112x62x30 mm).

Now it is possible to access *any number* of Centronics interfaces. These are used for fast data transfer to parallel printers or, Lap-Link style bi-directional communication. *Unlike some*, these interfaces are implemented to the - full Centronics Standard, i.e., all error, control, and data lines are connected. All are accessible from software. Like other network interfaces, network printers can shared by all linked computers. Using Amadeus, the QL is quite capable of printing to more than one printer at the same time! The cost for these network interfaces?, just £35.00.



Now another first for the QL, Record and Play back sounds via you computer. Our brand new product, Ama-Sound, is capable of recording and playing back sounds via any networked computer. Recorded files may be edited, stored and replayed. Complete with Microphone, Speaker and software, this interface at just £49.50, represents exceptional value for money.

Now Transfer data between connected computers at high speed. Over seventeen and a half thousand bytes a second are transferred between a Gold card QL and 486 PC!, *Impressed?*, well try it with a Super Gold Card! The speed of transfer on a Trump Card QL roughly equates to 3.5" disc drive transfers.

TRIALS TABLE	Bits/second	Bytes/Second	RS232 equivalent
Trump Card QL - 386 PC	40000	5000	55000bps
Gold Card QL - 486 PC	142000	17750	195250bps

Trials carried out with 125,000 size file being transferred from QL RAM drive to PC RAM drive

Soon Amadeus Protocol advanced inter-computer data transfer and file handling software will provide Amadeus with a highly advanced file serving and data transfer capability. All computers on the system will be able to; access each others disc drives and devices, send messages, handle remote text screens, etc., regardless of machine type.

Soon Fast RS232 interfaces capable of up to 115kbps will be available.

Soon An interface, especially designed for other manufactures or DIY enthusiasts, will be available to provide links for adding your components to the system.

Projects: Fileservers, memory buffers and many other useful interfaces are planned.

Amadeus Interlink has significant implications for the QL. Not only is it possible to link QL's to any number of I/O interfaces, but linking to other types of computer has also become a viable reality. Currently, QL's and PC's are able to access the system. Over a period of time, many other popular types of computer will be linked. Amadeus provides the ability to maximise resources, easily, and most importantly, *at low cost.*

Potential users will obviously be interested in the network capability. Amadeus is capable of shifting data around the network at about 2.5 Mbps *(unlike some networks there is no usage degrade)*. This may not sound very fast, but when you consider that 2.5 Mbps stands for 2.5 million bits per second, it will become apparent just how powerful the system is, (compare it to your 9600 bps RS232 baud rate)

Low Cost networking, from Di-Ren

Fleet Tactical Command II, QL & PC Versions
Users, please note this programme will be updated in the *near* future to allow usage over the Amadeus Network.

Contact Di-Ren for products catalogue

CARDS ACCEPTED - ACCESS/MASTERCARD/VISA Etc

Di-Ren Working for QL s since 1987

Telephone 01922 33580
Monday - Friday 0900-1700 (UK Time)
Answer phone sometimes enabled

⇦⇦**For Complete details contact**⇨⇨

Di-Ren
59 William St
Walsall, West
[redacted]
WS4 2AX, England

NEW (EEC) W.N. Richardson & Co.

Telephone/Fax 0494 871319

SINCLAIR QL PRICE LIST AUGUST '94

6 Ravensmead
Chalfont-St-Peter
Buckinghamshire, SL9 0NB

WE NOW STOCK CAMBRIDGE Z88 PORTABLE AT £99 AND CARRY A FULL RANGE OF ACCESSORIES

S.A.E. for further info. on hard disk drives, floppy disk drives, Z88 and Falkenburg products and accessories

BACKUP QL WITH P.S.U. JS ROM - £80 JM ROM - £70
PART EXCHANGE ALLOWANCE - £15

Accessories

* NOTE: EXTERNAL (SER2) 3 BUTTON MOUSE AND SOFTWARE WITH EXTRA FUNCTIONS. NOW HERMES COMPATIBLE. *

PC KEYBOARD INTERFACE, for INTERNAL FITTING, with lead to 102 KEY KEYBOARD		£ 75.00
PC KEYBOARD, UK version, 102 key (AT)	REDUCED PRICES	£ 25.00 ~~£ 30.00~~
* PC KEYBOARD INTERFACE and KEYBOARD		£ 95.00 ~~£ 99.00~~
CASE and LEAD for EXTERNAL FITTING of Keyboard interface for simpler assembly		£ 14.00

SER MOUSE, 3 Button, Software controlled, with operating and formatting disks externally mounted, fits in SER2 £ 40.00

TANDATA MODEMS — *1Mb, 2Mb and 4Mb* — *AGAIN IN STOCK*

ECONOMY MULTI-COMPUTER FLOPPY DISK DRIVE

NB: 4MB DRIVES NEED GOLD CARD OR SUPER GOLD CARD

CAPACITY:	1Mb DS/DD	2Mb DS/HD	4Mb DS/EHD
SINGLE DRIVE	£69	£85	£120
TWIN DRIVES	£99	£150	£199
10 VERBATIM DISKS	£5	£8	£35

TELEPHONE RE: UPDATING EXISTING DRIVES. ALSO DETAILS OF HARD DISK DRIVES.

Monitors

PLEASE ENQUIRE FOR COLOUR AND MONO MONITORS £40-£150

Microdrive Cartridges and Spares

4 New Cartridges in a wallet	£ 10.00	8 prog carts for reformatting in 2 wallets	£ 15.00
Plastic Storage filing box including 20 new cartridges			£ 45.00
QL Psion Software V2.35 Includes Quill, Abacus, Archive, and Easel IN WALLET			£ 18.00
QL Psion Software Separate programs			£ 10.00

QL power Supply Unit	£ 10.00	QL Printer I/F	£27	Membrane (and instructions)			£ 9.00
TV or Network leads	£ 3.00			QL Top & Bottom Case			£ 5.00
ICs ZX 8301	£ 9.00	ZX 8302	£ 3.00	8049 (IPC)	£ 3.00	MC 1377	£ 3.00

QL SERVICE MANUAL & CIRCUITS £25.00

S.A.E. FOR FURTHER DETAILS

VISA

Payment terms:
CWO, Access, VISA et cetera
Cheques - allow 10 days

Delivery:
Carriage - £9
Postage - £5

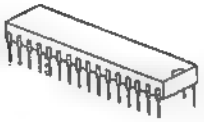
Access

Product subject to availability, E&OE

TEL/FAX: 0494-87-1319

MOBILE: 0850 597650

The STORY of "RAM"



Zapresic, CROATIA - Zeljko Nastasic

Anybody who is into computers these days is bound to hear things like dynamic RAM, cache RAM, video RAM, static RAM, dual port RAM. Almost anyone knows about RAM, but what are these strange words in front? What do they all mean?

Most of today's computers (which come in many forms, from the chip in your TV remote to super computers) use RAM for data storage. I won't go into why it is used, that was excellently described in the 'Computers 101' series.

Here's how the story goes: Once long ago (in the dark ages, as computer evolution goes) RAM memory used to be mechanical. Then, soon enough it was made out of some very special vacuum tubes - that was around the time J. Von Neumann first described what is today thought of as a computer. The tubes soon got displaced by transistors and magnetic cores - long before people even got the idea of chips. This was the first real RAM, in the sense we think of it today. It was complicated, made of thousands of tiny (0.3 mm diameter) ring shaped magnetic cores - one core for each bit of memory - and hundreds of meters of wire. It was relatively slow, but also maintained it's contents when power was switched off. The main disadvantage was that it was extremely expensive, because it could not be made by machines. Then, when the chips came into being, it took only a few years for the first RAM chips to appear.

The RAM chips used a circuit called a flip-flop to implement each bit of RAM memory. At first, only a limited number of components could be put onto a single chip, so the capacity of each memory chip in bits was limited - a very far cry from today's megabytes, the first RAM chips held only 16 bits of memory. But it was a start.

Those first chips, using flip-flops, were what is today known as STATIC RAM. Of course, then it was known as more or less the only kind of chip RAM.

Static RAM - but not for long!

Static RAM, or SRAM for short, needs only the availability of power to be able to maintain data stored into it. That's where the name comes from - if you want it to simply keep the data, give it power - and leave it be (= leave it static). SRAM is fairly expensive per a bit of storage, because the circuit used to store data is relatively complicated, using 4 to 6 transistors for each bit. The up side was that when it was competing with magnetic core memories, which were in wide use when the first SRAM chips appeared, it was much faster than magnetic cores.

As technology became better, more and more bits could be squeezed onto a single chip, so they soon became cheaper per bit of storage. But one big problem remained - in order to keep it's contents, a static RAM had to draw current from it's power-supply - even if nothing was written or read. This was completely unlike the magnetic core memory, which uses power only when it is actually accessed. Soon it became obvious that the new SRAM chips could not compete with large magnetic core memory, because although the chips themselves were cheap, the power supplies and various cooling methods which had to be used because many chips carrying many transistors were crammed in a very small place, soon became more expensive than the memory itself.

This is why SRAM traditionally became used where speed was the main concern, and where smaller amounts of memory were needed, such as in processor registers (remember that at the time processors were made of many simple chips - the microprocessor wasn't invented yet, because there was no technology to make it possible). In an attempt to find a solution to the power requirement problems, the chip designers of the time stumbled on a principle which is used in the vast majority of today's memory chips - the principle of the dynamic RAM.

D is for dynamic: Dynamic RAM, or DRAM was invented practically by pure chance - which is all the more strange, when one considers that a vast majority of RAM memory in existence today is of that type.

As said earlier, DRAM was a product of designers trying to lower the power consumption of early SRAM chips. Legend has it that someone in the Fairchild company (which was the first manufacturer of chips altogether) discovered DRAM. But because it's a legend, it only means no-one really knows to this day who invented DRAM.

The STORY of "RAM" - (CONT'D)

Anyway, one of the computer designers accidentally learned that when SRAM was switched off, it's contents still remained there for a length of time, long after no more power is given to the chip. Very soon after that the first DRAM chips appeared, which were very different than today's - they simply periodically switched off their power supply when they were not accessed, in order to reduce power consumption.

In an effort to find out what was causing this behaviour, researchers soon discovered that certain structures on the chip behaved as capacitors. Somewhere around that time the first MOS transistors could be manufactured on a chip - and a capacitor is an essential part of a MOS transistor. Capacitors have the ability to store electrical charges. Someone quickly put two and two together - and got the first real DRAM.

A DRAM, like SRAM, has to have a power supply to function, but unlike SRAM, data that is stored into a DRAM vanishes after a certain time. In a DRAM one capacitor is used to store each bit of data, so the data is represented as the charge stored in the capacitor. Because no capacitor is perfect, the charge slowly dissipates, and when it dissipates completely - the data represented by it is lost.

What's the point then in having a RAM that forgets? Well, as long as you periodically 'refresh' the contents (by reading them and writing back with a full charge) the data will be retained! This is why DRAM is called dynamic - it needs something done to it to retain the data, if you want it to store data. If it went 'Static' it would forget all the data in a fraction of a second.

At first this dynamic thing seems to be a serious drawback - the power reduction alone couldn't justify it. The real reason why this peculiar habit of the DRAM is tolerated is that it uses only one transistor per a bit of memory, and can thus have at least 4 times the capacity of a SRAM using the same manufacturing technology. And because the price of the memory chip is directly proportional to the number of transistors it carries, it soon comes out that for a same price you get at least 4 times more memory if you use DRAM instead of SRAM. At the time this was a HUGE saving, because it also saved on power, and the first DRAMs were even faster than SRAM because of the reduced number of transistors per bit. Although the power problem in SRAM found it's solution in due time, one DRAM advantage remained - that is the vastly larger memory capacity you get for the same price. This is why almost all of today's computers use DRAM chips for RAM memory.

What also remains is the slight drawback of the DRAM needing to be refreshed. I say slight because in today's DRAM chips this is a relatively rare event, so it does not use much time. In fact, DRAM refresh uses up about 0.5% to 2% of total memory accesses in modern designs, which can freely be dismissed. The added circuitry to do this has also became much simpler with time, because much of it was integrated onto the DRAM chip itself. Currently available DRAMs have a capacity of up to 16 megabits (2 megabytes) per chip, and the access takes from 50 to 100 ns to complete. DRAM chips typically cost some £30 or less per megabyte. Finally, 64 megabit DRAMs are expected to appear very soon - cramming up to 8 Mb on a single chip.

As can be seen, with the improvement of technology, DRAM characteristics have also improved - to the point where it's need to be refreshed isn't any longer a major concern. Because DRAM has to be refreshed, and is not static, it does not have the advantage of modern SRAM that almost no power is used if no accesses are made, because accesses have to be made for refresh - but more on that later.

Unfortunately, one much more serious weakness remains - as designers tried to squeeze more and more bits on a DRAM chip, today's DRAM's capacitors which store the bits have such a minuscule amount of charge that even reading them completely uses it up. Because of this a DRAM includes circuits which make a copy of the data when it is read, and write it right back again so it isn't forgotten. This same process can be invoked using a special sequence of signals to the DRAM chip so that DRAM contents are refreshed. But because once data is read it has to be written back again, another access cannot be performed until this is complete - it takes the same time as the normal access. Also, before data is written, old data has to be 'destroyed' so that it does not interfere with the new data - this is again done by reading the old data to destroy it. Although this two-step process cannot be seen from outside the chip itself, it makes DRAM twice as slow as SRAM, because SRAM only needs one step to complete an access.

In effect - DRAM is two times slower than SRAM manufactured using same technology. This soon became a really big problem - the speed of the RAM of a computer became the bottleneck, and you could not use SRAM because it

The STORY of "RAM" - (CONT'D)

was too expensive for the RAM size needed. A solution was found by a few very clever people at IBM - strange for a company which is considered by some very traditionalistic in it's views, but that's how it was - and it was called the CACHE.

Cache - the static RAM strikes back: Cache RAM is only one part of the cache circuitry, albeit an important one. It is usually a relatively small amount of fast SRAM, which is in many of today's CPUs integrated onto the CPU chip itself.

Cache was first introduced in the IBM 360 series of computers, along with an inordinate amount of new and clever ideas (virtual memory, memory interleaving, pipelined execution of instructions, etc). Engineers at IBM wanted to somehow simulate fast memory by using a buffer made of fast SRAM, which was filled in advance with the data from the slower DRAM. As long as the data the CPU needed is already in the fast buffer, it could be accessed fast. But if it was not, it would take a longer time, because the CPU would have to wait for the data to come from slower DRAM. Also, when data was written, it had to be written to both the fast and the slow memory, so that they would always be coherent - meaning their contents would always be a copy of each other. This meant that write was always slow.

The ability of such 'compound' RAM to behave as fast SRAM was obviously as good as the logic used to get data from the slow DRAM into the fast SRAM in advance - this is today known as 'caching algorithm' or 'caching strategy'. The big problem here is how to know what data the CPU is going to need in advance - when the CPU itself does not know that until it actually has to fetch this data.

The CPU mostly spends time reading instructions, less often reading or writing data as a result of the instruction execution. Some clever thinking soon pointed out that when the CPU performs a jump instruction to change program flow, it very often jumps backwards a number of instructions - not far at all, to execute a loop. This became known as 'locality of reference'. On the other hand, it was much more complicated to predict what data the instructions executed would need. Therefore the first IBM 360 machines used a cache only to speed up access to instructions in RAM.

As the CPU worked, every instruction it would read got stored into the cache buffer - which in effect held a 'history' of the instructions executed. How could this help when you had to wait for an instruction to be read from slow RAM in order for it to be placed in the cache in the first place, you might ask - but this is where the locality of reference comes into play. Every time the CPU would jump back to an instruction it already executed, for instance when it was running in a loop, there was a good chance the instruction was already in the cache - and then the CPU would not have to wait! Only if it jumped forward or far back was waiting needed. But as soon as it started doing a loop - the cache would show it's benefits. This was the first cache strategy ever devised - it was a simple one which needed a bit more logic, but the minimum of fast RAM, and it worked magnificently - the IBM 360 range of machines was a success, and was redesigned and improved many times - and many still operate to this very day.

Today no-one would consider such a 'primitive' cache (although I've heard rumours that Sinclair's PGC 76C00 RISC processor has something very similar) - but then it is easy to be smart today! In due time strategies were devised which allowed data to be cached and not only instructions, and be able to keep the data and instructions needed by the CPU very cleverly, so that the CPU does not wait for the memory 80% of the time, and sometimes approaching 100%. It can be argued that there are as many cache techniques and strategies as there are computers, which work with more or less success. Research showed that even small amounts of cache can bring significant performance improvement if the cache strategy is chosen well - and on the other hand, when the cache size reaches a certain limit (which is far less than the total amount of RAM) there is no more improvement in speed.

One notable example of efficiency is the cache used in the 68020 CPU - the very CPU which powers the Super Goldcard. The 68020 was the very first single chip microprocessor (yes, there were multiple chip microprocessors, and there still are) which had an integrated cache on the chip - again only for instructions. The first computer ever to use a cache which works the way a 68020 cache works, is the Cray 1 super computer - in fact the similarity is stunning, right down to the amount of cache memory used. The 68020 cache also works by storing every instruction the CPU reads - but it also stores an address where this instruction was found, in a very simple way. Because of this, it can store much more frequently used instructions, even if they are somewhat scattered over the RAM.

The STORY of "RAM" - (CONT'D)

Although the 68020 has only 256 bytes of cache, and an exceedingly simple caching strategy, it is very efficient - Eros Forenzi of QItaly has shown that using the cache can boost performance by 40 to 80 percent on typical programs.

In contrast, many 80386 machines manage less with 32 kb of cache (sometimes even more, 64k, 128k or 256k) - clearly emphasising the strategy used and not the amount of cache RAM. In time caching improved significantly, so much so that today no-one even considers a fast CPU without an on-chip cache. For instance, the 68040 CPU used on the QXL has two caches, one for instructions and one for data, 2 kb each - adding significantly to it's performance.

It was also discovered that the cache philosophy can be used for other things a CPU does, so for instance the 68060, which is currently the most advanced of the 68000 series processors, has no less than 5 caches.

Wheels within wheels within wheels... History has a habit of repeating itself, so some CPUs are so fast, that even with internal cache, the demands for external RAM speed can be far larger than simple DRAM can manage. To help matters in such cases, some designers use yet another cache memory between the internal cache of a CPU and the DRAM. This is called a second-level cache. Sometimes this is used if the internal CPU cache algorithm is not efficient enough for the typical applications being run on the machine - mostly when they waste memory, like Windows programs on the PC (Me to think of criticising the PC? - Nooo!).

In some very special memory topologies, which emulate very large RAM by using hard discs (this is called virtual memory, and it's another story, OK?) there can be a further cache level between DRAM and the hard disc, called a disc cache. This technology has since emigrated right into perfectly 'normal' computers and hard discs, which was really to be expected - since hard disc is a form of RAM memory (a somewhat peculiar one), it turned out that a modification of the 'locality of reference' principle can be applied - only instead of frequently used memory locations we play with frequently used files, or parts of them - like a frequently used part of the directory. Therefore, caching can, and does work. But if you thought that was all, don't worry - it's not.

Some of the newest processors still use ANOTHER (Aargh!) level of cache, this time between the 'real' cache and a part of the processor known as the execution unit. In a CISC (as opposed to RISC) CPU, before an instruction goes into the execution unit it has to be converted to an internal format instruction or instructions - known as microinstructions. This special cache caches microinstructions, to avoid locality of reference in the normal instructions make the converting circuitry convert frequently used instructions over and over again.

Batteries not included: Were it only for the use of SRAM in cache memory, there would have been no need for SRAM sizes we have today. The turning point happened when a new technology was introduced in the early 1970's, called CMOS. This technology uses a different type of transistor, and a different circuit technology, which makes it use power only when the logic level is changed, from 0 to 1 or 1 to 0. Between changes it uses almost no power at all - which had tremendous ramifications to almost all digital circuits - for instance, the reason that a Goldcard or a QUBIDE consume very small amounts of power, lies in the use of chips manufactured using improved CMOS technology. It can freely be said that CMOS made laptop computers possible.

To get back to the story - soon enough, the first CMOS SRAMs appeared - and they were a smash hit! Very soon they established themselves in low power devices of all sorts - especially where battery power was used. This generated more sales and then prices slowly went down. As the prices went down, SRAMs found use where they would not have been used before - because of the far lesser power demand, savings on the power supply were more than the added cost of SRAM, and further, because SRAM is easiest to use, requiring no refresh, it became a must in all sorts of small computers, especially in the ones which are not visible, but control other devices - from car ignitions and dishwashers, to printers. Microcontroller chips (like the 8049 IPC in the QL, or PIC 17C42 on the new superHermes) which have a CPU, RAM, ROM and input-output on the same chip use SRAM for the on-chip RAM. Also, because of the way the circuit used to store information in a SRAM is constructed, it is much more resistant to all sorts of interference and noise, and even radioactivity, which makes it's use obligatory in high reliability equipment. In some cases large amounts of SRAM chips are used instead of magnetic memory (floppy and hard discs) either because of speed, or because of reliability.

The STORY of "RAM" - (CONT'D)

Today's technology can squeeze 512 kb of SRAM on a single chip. 128 kb SRAMs are commonly available at relatively low prices (some £10 or less). The time it takes to read or write data for current SRAMs is less than 100ns - less than one tenth of a millionth of a second. Top speed SRAMs can work tens of times faster. SRAMs with low power consumption can maintain their contents for over a decade running on a single alkaline battery - in fact the battery itself deteriorates faster than the SRAM can exhaust it. This has lead to some manufacturers putting a battery into the chip case - the SRAM is likely to be replaced or discarded as new types become available, long before it exhausts the battery. One thing can be said about today's static RAMs - when they are static, they are VERY static indeed.

The need for speed: Sometimes even caches cannot provide the raw speed needed - mostly because the memory is not being accessed by a CPU, so there is no 'locality of reference' which makes the cache do it's work. This happens when successive data is needed, for instance. If you remember, when I explained (or tried to) how the cache works, I mentioned that the CPU most often executes the following instruction after the one it just has, except when it executes a jump instruction. In a way this is a special form of locality of reference - one in which the following memory location is most likely to be accessed. However, it's quite different from the idea the cache uses, covering already executed instructions. Again, this new type of 'forward' locality of reference was exploited first by the IBM 360 - by interleaving memory access.

In essence, all even words were stored in one memory, and all odd words in another. Then, as long as the CPU executed successive instructions, this would happen - while the CPU read the odd word, the even memory would recover from the read made before that. Then, when the CPU went to the next word, the even one, it would be read from the even memory, while the odd memory was recovering (remember the two step access and recovery process in DRAMs?). This could go on at double the normal memory speed as long as instructions were read sequentially. When not - the CPU had to wait again. Many computers still use this technique along with others to improve performance when an instruction or data is not in the cache - this is called a cache miss.

Another example where neither caching nor interleaving is of much help is memory which is used to store a representation of a graphical screen, frequently called screen or video memory.

Consider a graphics screen with 1024 x 512 resolution, in say 4 colours - like that of the Miracle Masterpiece card. This takes 1024 x 512 x 2 bits of memory, which is 128 kb. Because the way display screens work, the contents of this memory have to be output to the display over and over again, the faster the better - on the Masterpiece some 50 times a second, which is referred to as the refresh rate. This is because the screen itself behaves as a sort of dynamic RAM, which has to be refreshed, otherwise the picture would fade away or flicker. In the case of the screen, there is no way to make the screen refresh itself on command, like DRAM does - the data representing the picture has to be rewritten over and over again.

If we use this fact to calculate how many bytes get transferred to the screen in one second, for the Masterpiece we get 6.25 Mb. Given the speed with which today's DRAM works, this is easily achievable - but there is a VERY big flaw in this calculation - the CPU or a graphics processor also has to access this memory, and preferably at full speed, to do the drawing!

Clearly this is a problem - firstly, this double access completely shatters all locality of reference, and what is much worse, we would need a memory which can be simultaneously read or written by the CPU and read by the screen refresh circuitry.

Such a memory is called Dual Port RAM, or DPRAM. In essence, it has double sets of signals for addresses and data, so it can be accessed by two devices at the same time. Unfortunately, such memory uses even more complicated circuits to implement a single bit, so has much less capacity than even SRAM, and a correspondingly higher price. Because of this it was never really considered for screen memory, and it's rarely used, so it never achieved very high volume production and the associated low price. Today DPRAMs with more than 16 kb of storage are very few and far apart, and extremely expensive. However, they are used in some special areas where nothing else will do, for instance in multiprocessor machines, for communications between CPUs. To give you a feeling of DPRAM price - you get a miserable 2 kb for £10! Clearly another approach was needed. Since screen refresh always accesses data stored in a succession of addresses, designers first cascaded memory chips to form memory of great widths,

The STORY of "RAM" - (CONT'D)

sometimes up to 256 bits - which allowed the refresh circuits to access 256 successive bits at a time, and thus appear to be very fast for successive data. However, this was soon proved inadequate as graphics resolutions and numbers of colours continued to rise, because the number of chips and the sheer space needed to carry wires for more and more bits became too large - something new was needed again.

Since DRAM is internally organised into very wide memory which only pretends to be narrower, but larger in length (number of addresses), someone probably thought - if only there was a way to use this wide memory directly... Since the vast amount of data that goes to the screen gets serialised (like serial ports, but many times faster) to be usable by the display, someone thought of putting the serialising circuit into the memory chip - in effect emulating the previous very wide RAM designs on the chip itself, with only little added cost - the wide RAM was already there. That's how the Video RAM, or VRAM was born. The secret of the VRAM is that it internally transfers one whole line of pixels (usually 1024 or 2048 bits) into the internal serialiser, which takes a single access. Then, until this line is exhausted by sending it to the display, the memory can be used much as a normal DRAM is by the CPU, or a dedicated graphics processor. In practice, these transfer accesses are very rare compared to the normal ones - generally they slow down the CPU or graphics processor access as much as DRAM refresh does (which is very often done at the same time - to save time), well under 2%.

One might ask - why then did the original QL have standard RAM for screen memory, in fact the very same one used as ordinary memory? The answer is simple - the resolution and number of colours were such, that when accesses were divided between the CPU and the screen refresh circuits (located in the 8301 ULA), the CPU still had 50% of accesses available - clearly an acceptable percentage for people at Sinclair Research. Actually, they are not to be blamed - an alternative solution would probably have cost more, especially in view of the fact that VRAM has just been invented at the time.

Today's processors are much faster, and the graphics needs much more memory - this has gone so far that sometimes even the requirement for screen refresh cannot be handled by standard DRAM, let alone having the CPU read and write to it at the same time. One notable exception are many PC VGA cards, which still use DRAM. However, CPU access to a VGA card screen memory is very slow - almost ten times what it could be. Because of this cards use very special techniques to allow fast access for the CPU, and still maintain high resolution and a large number of colours (but are still slower than cards using VRAM), like having a cache for CPU accesses, and implementing shadow RAM.

Into the shadows: The concept of shadow RAM is a well known one - but has very seldomly been used until prices of memory became low enough. This is because it involves using two memories of a certain size to impersonate one memory of the same size, but with some special facilities. One such use is 'shadowing' screen memory, something which is also done on the Goldcard and Super Goldcard, to improve screen speed.

When the QL CPU writes to the screen memory, it will get slowed down to half speed, because of the screen refresh demands. The same will happen if data is read from the screen memory. Taking into account that only the write is actually significant, as the CPU can only expect to find what it has written itself - the screen not being able to work 'backwards', making a picture into bits of data - we soon see that there has to be a way to at least make reading fast. This is accomplished by holding a copy of everything written in another memory - the shadow memory.

When the Goldcard CPU writes to screen memory, it will simultaneously write to the shadow memory. The slower memory determines the speed of the combination - and this is the QL screen memory. However, when the CPU reads from the screen memory area, it actually reads only from the copy in the shadow memory - which is much faster, on the GC at least 4 times. This makes the screen faster, because in order to put dots onto the screen, the CPU has first to read a byte or word from the screen memory, change only the bits which represent the pixel to be changed, and then write the byte or word back again. In essence, data is read from screen memory quite frequently, which greatly benefits from the shadow memory scheme. Shadow memory can also be used for other things, as long as the 'real' and the shadow memory are always a copy of each other.

RAMifications: All these types of RAM are constantly being improved - to get more speed, more capacity, or special abilities.

The STORY of "RAM" - (CONT'D)

The consequence is that the immediate future holds some very interesting RAM architectures - for instance EDRAM, or Enhanced DRAM. Actually, it's only DRAM, but someone finally got the idea to integrate some cache onto the same chip. The result is DRAM that almost always behaves as fast SRAM.

Another very specialised RAM is SLAM, or Scan Line Access Memory, a kind of RAM very similar to VRAM, and is like the latter used exclusively for graphics. It again exploits the internally wide architecture of the DRAM to combine VRAM circuits, and a special circuit which enables filling memory which represents part or a whole line of dots on the screen with a pattern, in a single access. This can be used to draw filled polygons extremely fast, literally billions of dots in a second, for the purpose of generating animated graphics. Such special memory is used in very high performance graphics computers, for instance in flight simulators.

However, with the advances in technology, don't be surprised if you find them in your computers, and worse still in games consoles! So, until we all go on to using something completely different, there are a lot of other designs and strange abbreviations to be seen...

THOUGHTS ON QL REPAIRS

Ascot, Berks, GREAT BRITAIN - Tony Firshman

Those were interesting articles by Dennis Briggs and Don Walterman in the last issue of IQLR on QL repairs. Here are a few comments based on some 1300 QL repairs for customers.

I always swap in good chips when trouble shooting, and have had no problems (despite what Dennis says!). Also the legs are so malleable it is very difficult to break them. The legs are worth cleaning, especially 8302. I use a hard rubber or the special fibreglass pen when I can find it. JS 0000 pin 1 is often missing - this is normal (anyone know why?)

First of all though, try the BARE QL and a different monitor/lead (or TV) and power supply that are known to work. Amazing how often I get fully working QLs to 'repair'!

Always check the expansion connector (J1). It is very easy to bend pins on this if expansion cards are not inserted square. Usually it is just one of the spare GND or 9v lines bent to the plastic surround. Sometimes though one of the signal lines gets bent against another (or 9V!). This usually stops the QL from starting up. Be VERY careful when straightening. I usually remove the nearest circuit board fixing screw (and the rom cover), prop up the end of the circuit board, raise the pin with a small screwdriver, and then straighten and align it with long nose pliers. Beware - the pin is very brittle, and it will not survive another bending!

Problems found in order of frequency:

1) MEMBRANE/KEYBOARD The membrane becomes brittle with age/heat. The tail ends can be cut if they break. The usual break though is where the wide tail (nearest heat) bends. Also most IBM keyboard interfaces fall out of the 8049 socket eventually - esp schoen. superHermes will be quite a small board and should NOT suffer this way. If you have a substantial CLEAR membrane then cheer - you have an early one which survives. I guess Sinclair dropped it 'cos it cost 10p more.

Don't connect a screw to the hole below space bar. This hole is only there on early QLs, and can cause spurious spaces.

Often repeated chrs (esp C/V/B etc or one of these when space pressed) is often due to dirt on membrane/bubble mat.

2) 8301 Commonest causes of failure are monitors (esp Microvitek cub - where there is often a HV crack) and Gold Card. The latter is probably due to different GC power up rate. Often I see them go when there is a mains 'brown-out' (voltage drop). You have 3 seconds to switch off before 8301 really BURNS! Miracle changed a ceramic cap to electrolytic on GC. Super GC seems fine.

THOUGHTS ON QL REPAIRS - (CONT'D)

Dennis talks about the modified ceramic 8301s. Maybe these are the 50 I made for Syd Day in 1988 or so and mounted a packaged 15 MHz oscillator on top. This was when 8301s ran out. Unfortunately (for him) as soon as I had done them at around £6 each, supplies came back. It was sad to see most of these in his garage after his death.

3) RAM I of course also use Minerva ramfail_bas. I manage to replace the chips easily. They are not straightforward because the pins are bent onto the pads. I use a purpose built electric blower with a stand (but a B&D paint stripper would be fine!). I heat the chip from underneath, pull out with an extractor, reheat the pads and drop in a socket. It all takes 30 seconds. It needs a LOT of practise though to avoid overheat. The only failures I have had (popped vias) can be cured by soldering a wire through.

4) SERIAL Order of frequency:

. 1488 (and combination with 8302 - some 8302s have low output)

. power supply ac

. +12v (yes some 7812s get quite hot, and are UNDER heatsink!) Issue 6 boards usually had a bigger 7812

. -12v

. 1489 (one or two only)

Serial is difficult to pin down. I use a loopback lead (two connectors wired together - 1-1, 2-2 etc), and test loopback. If this fails and all voltages are OK I do:

```
open#3,ser1i:rep loop:print#3,'U'
```

and then with handshaking and then for ser2, and look at all signal lines with scope.

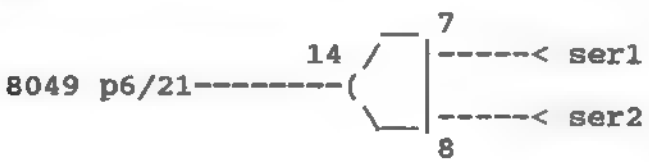
Sinclair test is:

```
OPEN#3,ser1:OPEN #4,ser2
PRINT#3,'Test':PRINT #4,'Test'
INPUT#3,in1$:INPUT#4,in2$
```

or similar, at all baud rates with loopback lead. This tests handshaking too of course.

I find that the two flying resistors (33k from 8302 p19/21 to -12v) are often needed to get serial loopback working. Odd 'cos they are connected to mdv lines. There must be some internal effect - maybe changing output levels.

Dennis doesn't point out the serial input buffering on hal (issue 6):



5) MICRODRIVES USE FLOPPIES!! Often the rubber roller will rise up. Check the motor screws. Ribbon cable can work loose. 78M05 regulator legs can break - resolder. If mechanically OK, and doesn't run, try changing TR4 or TR5. If all seems OK and will run but not format on a NEW cartridge, then probably the ULA. This CAN be replaced (needs cutting out) but not really worth the trouble - get a new mdv unit.

6) POWER SUPPLY/regulator: Often the symptom is slowly scrolling ripple - which can be a sawtooth power supply, but can also be a failing 7805.

THOUGHTS ON QL REPAIRS - (CONT'D)

I must say I have only experienced maybe 3 p/s with sawtooth that Dennis describes. In these cases the variation of the DC was from 9v to 10.5. I have never seen the 7v to 12v he presents as the norm. Most QLs I have checked have a fairly steady 10 to 11v - hence the high temperature of the 7805. I find as long as the 7805 is in good condition, and properly screwed down with heat sink compound, there is no 'overheating problem'. There has been so much emphasis put on this, but I found way back in 1985/6 that most times locking up due to 'overheating' was cured by using a mains filter - ie NOT overheating. These MUST be used with standard QLs. With GC/Super gold card the situation is much better, especially if the redundant high current 68008 is removed. Oh yes - I had one QL for 'repair' with no 68008 - yes you guessed it!

7) 1377. This is thought of as providing UHF or composite video. However on some monitors it provides vertical synch, so if the screen of a monitor is scrolling, check +12v and 1377.

Oh yes, number ONE should really be a mechanical check - ie monitor/serial leads, screws in mains plug (yes they can even fall out!), bent expansion connector pins and so on.

8) 8302 (rare) Often a failed 8302 will simply mean ram check and no F1/F2 or more commonly no reaction to F1/F2

9) 8049 (rare) Usually means ram check but no F1/F2

First thing to do if you get ram test but no F1/F2 is to remove 8302 and 8049. If still no F1/F2, then it is probably ROM 8000.

(Oh yes - best way to bypass F1/F2 is not, as Dennis suggests, blow a special rom, but to use a joystick! [or Minerva which presses F2 for you of course!] I use a switch connected to a ctrl plug)

10) NETWORK often fails cos terminator (empty socket) has failed. Bending centre pin connector back on empty socket CAN work, although difficult. Best solution is to have two jack plugs with 330 ohm resistor soldered inside).

LED wire colours on some replacement keyboards are different. The std colours are red(power) black white(mdv1) black grey(mdv2) black from the back of the QL. Of course the order of the black (GND) doesn't matter.

Board repairs are very time consuming, and I must admit if it takes any time, I usually substitute a working board, and save the old one for a rainy day! Maybe I should give them to Don. Mind you I guess I have only had to bin some 20 boards over the years.

The piles of faulty boards I got from Syd Day though are another matter. Most of them are used for lifting parts (ram etc).

I think the repair I enjoyed most was for a Eggert Peterson in the Middle East. He faxed on Thursday. In view of his comments, I suggested we use datapost. The QL arrived late Friday, I returned it first thing Saturday, and he had it running I believe on Monday! The one way shipping cost alone was about £45, but he was very pleased!

The worst was for a client in Eire who shall remain VERY nameless. The QL arrived via another trader unboxed and no power supply. I repaired it (new circuit board), charged repair without VAT and added the Eire shipping cost. It came to about £3 more than a 'normal' repair, but that was all shipping. He did pay this. I suggested the fault was due to something outside the QL - say a fualty power supply. The QL arrived back in three weeks with the same fault (ie completely unrepairable main board) - I replaced the board again with no charge under guarantee. Bear in mind that by now I have scrapped two main boards and paid £9 extra return postage. It was back again in a month with another QL, both with failed 8301s. He said that as the FIRST QL was clearly unrepairable, he required a replacement QL. He also subtracted £14 or so from his enclosed cheque for the cost of the two mailings of the original QL. Hurrmm! I repaired both QLs with 8301s, and waited for his full payment for the second repair (some three months later and many long scrawled/illegible/abusive letters!) before returning both. I haven't heard direct from him since, but I have heard stories from others of him saying what a shark I am. ***Those were the days.***

TF SERVICES

MINERVA

The ULTIMATE system upgrade

MINERVA RTC (MKII) + battery for 256 bytes ram, CRASHPROOF clock & I²C bus for interfacing. Can autoboot from battery backed ram. Quick start-up.

OTHER FEATURES COMMON TO ALL VERSIONS

DEBUGGED operating system/ autoboot on reset or power failure/ Multiple Basic/ faster scheduler-graphics (within 10% of Lightning)-string handling/ WHEN ERROR/2nd screen/TRACE/ non-English keyboard drivers/ "warm" fast reset. V1.97 with split OUTPUT baud rates (+ Hermes) & built in Multibasic.

1st upgrade free. Otherwise send £4 (+£6 for manual if requd) . Send SAE, Minerva + disk or 3 mdvs.

MK1...£41(£40)[£43] RTC(MKII)...£66(£63)[£67]

GOLD CARD (incl SUPER) compatible

HERMES

A replacement QL co-processor for the QLs awful IPC 8049

- Do you get keyboard bounce?
- Do you find fast serial input unreliable?
- Do you want to connect a faxmodem at 19200bps and and send and receive FAXES and/or data.

If you can say one YES , then you need HERMES

- 19200bps RELIABLE serial input - NO QCONNECT.
- Independent input baud rates - use serial mouse & print
- Stops keyboard bounce (unwanted repeat chars)
- Improves 'fuzzy' and 'random' sound
- Provides extra input/output lines
- Key click

To fit, simply replace the QL 8049 or 8749 chip

£26(£24)[£27] incl manual/software

QL SPARES

Faulty QL board (no plug-in chips).....£11(£10)[£16]

Kybd membr £9.50(9)(10.50)	Circuit diags . £3.50(£3)[£4]
68008 cpu .... £8.50(£8)[£11]	1377 PAL ..... £3.50(£3)[£4]
JM ROM...£10.50(£10)(£11)	Power supply £17(£16)[£26]
8302 ULA...£10.50(10)[£11]	(220/240v) surface[£21]
8049 IPC ...£8.50(£7.50)[£9]	8301 ULA £10.50(£10)[£11]

Other componcats/(sockets etc) please phone

QBBS

UKs first QL scrolling Bulletin Board

Megabytes of files. Messages to/from UK/Belgium/ Holland/USA/Germany for a UK phone call.

TANDATA callers add SIX zeros (000000) or wait for 3 seconds of modem tone if dialling manually.

(+44)344-890987 (up to V32bis)

I²C Interfaces

The I²C bus was designed by Philips to simplify interfacing. Minerva MKII clock is driven by an I²C chip, & a connector allows connection of other circuits. Our external circuits will interconnect without leads. Up to 5 interfaces can be powered off the QL - up to 4 of each can be separately addressed from the QL. No soldering required to fit to the QL. High quality cases and labels, with professionally made double sided circuit boards. Interface sends & receives at 100k bps

Parallel Interface gives 16 input/output lines. Can be used for logic level output eg model train controllers. Input direct to motor drivers (eg L293/ L298).....£26(£24)[£27]

Analogue Interface Each gives 8 analogue to digital inputs, and 2 digital/analogue outputs. For temp measurement, sound sampling etc.....£31.50(£29)[£30]

Data sheets. (analogue/parallel chips).....£2.50(£2)[£3]

Control software/manual .....£2.50 (£2) [£3]

(First interface purchase includes free 15D/9D lead)

Prices include AIRMAIL post & packing & zero VAT rated outside the EC. Prices are: EC except UK (Europe) [outside Europe]. Ring for UK prices or see QL World/Quanta/QReview. Payment can be by Mastercard/ Visa/Access/Eurocard/£ cheque/UK postal order/money order/bank draft drawn on branch in the UK or CASH!. (NOT Eurocheques due to crazy bank charges - £15!!) MAIL ORDER ONLY - no callers without ringing first. Any customs duty etc customers responsibility. Send IRC for full list and details.

VISA

Holly Corner, Priory Road, ASCOT, Berks, SL5 7RL

Tel:(+44)1344-890986 Fax & BBS: (+44)1344-890987

MasterCard

SMSQ/E on the QXL

Santa Clara, California, USA - James D Hunkins

Tony Tebby's SMSQ/E
Q-DOS Compatible Operating System for the 90's and Beyond

"Tomorrow's Operating System - Today"

For the last several months, there has been a lot of press and discussion of SMSQ/E, Tony Tebby's latest serving of QL software inspiration (and sweat). In case you haven't had a chance to keep up with the news, SMSQ/E is a new operating system written by Tony Tebby to replace QDOS on any QL compatible hardware which has a faster processor (faster than the original 68008) and 1 Meg or more of memory.

SMSQ/E is QDOS 'compatible' and will run nearly all QDOS software. There are separate versions for the different hardware platforms (Gold or Super Gold card equipped QLs, Atari based QLs and the QXL card). This article will cover SMSQ/E on the QXL card. In addition to being of interest to QXL current and future owners, the review may be of interest to non-QXL owners, as the different versions of SMSQ/E are basically the same except for the required support for the different hardware.

One note for non-QXL users, please look for more information on SMSQ/E for your hardware. Many of the new capabilities in SMSQ/E already exist within the operating system shipped with the QXL card (SMSQ). Therefore, these 'pre-existing capabilities will be only briefly mentioned here, but definitely should be more closely examined if you are not already familiar with them.

As Shipped from the Factory - SMSQ v2.47 on the QXL

SMSQ/E is basically an expanded version of SMSQ, the operating system shipped from the factory with the QXL. In case the reader is not familiar with SMSQ (v2.47 as of this review), a brief summary of its capabilities over QDOS is appropriate before proceeding. All of the following features are also available under SMSQ/E.

SMSQ v2.47 is a major step up from previous releases. The following discussion is based on v2.47 and may contain some new items for those who haven't had a chance to play with it yet. Operating System Enhancements: first of all, it is fast. The QXL itself is already a fast piece of hardware. And SMSQ, through its carefully being written from scratch, makes it even faster.

A new job priority scheme is used to smooth out response to user input. The priority scheme can be reset to the original QDOS method if desired to push for a bit more performance. For those favorite games, the SLUG command is now included to slow the QXL so that the player will not loose before he/she even starts.

Several things come packaged with the operating system. All the TKII commands are included. Additionally many of the popular Minerva capabilities have been supported. These include expanded PEEK and POKE capabilities, such as the ability to directly access system variables. Many new ways to move through the edit line are also added.

One of my favorite changes is the ability to run BASIC programs with the EXEC type functions or from the file manager. For users of QSAVE and QLOAD fast loading, compatibility is now built in. For some of the file commands you now have a choice of using channel number or file names. For those who like hexadecimal or binary numbers, numbers can now be directly entered in these formats [$nnn %nnn] (a blessing for programmers).

For people using different languages or printers that require special characters, the TRA command has been 'enhanced' to allow different translation tables depending on language or TRA number assignment within a job. For high resolution display users, you can now offset the super basic windows with both WMON and WTV commands.

Hardware Support: Next to speed, the most visible capability of SMSQ is the usage of high resolution screens. I normally run my screen in 800x600 pixel mode and pack on many different windows at once.

You can of course access the parallel and serial ports. BAUD can now handle both serial ports independently. For programs that insisted on modifying their own code, the 68040 cache can be disabled and then turned back on for SPEED when you are done with the 'bad' program.

SMSQ/E on the QXL - (CONT'D)

To make sure that a program knows what hardware it is running on, system variables have been added to tell the machine and processor in use (on non-SMSQ and non-SMSQ/E hardware, these systems variables will equal 0 if they have not been previously set by a program).

Filing System: QDOS has always been easy to expand, especially when it comes to supporting other devices. SMSQ now comes with many filing system enhancements built-in. In addition to supportung hard disks (WIN_) and pipes (PIPE), SMSQ can read and write both QDOS or PC format disks transparent to the user, great for exchanging data between systems.

The DEV device has been added. It can be used to substitute for other devices or devices with directories. For example, let's say that I want to access programs in two different directories but the path names are very long. Device/directory names can be assigned to different DEVs (DEV1_ and DEV2_ for example). Appending the file name onto the DEV#_ now accesses the file. No more really long names to (mis)type.

Different DEV#_ can be linked, providing an enhanced form of the PC DOS PATH capability. If the program being looked for is not found in the current DEV#_, SMSQ will look at the next linked DEV#_, and then the next one until either the program is found or SMSQ runs out of linked DEV#_s to search.

For a bit of craziness, the NULL device is available. It is useful to QUICKLY throw away results or to make a program wait for input that will never happen. This may sound strange, but in addition to making for very fast benchmark numbers, it is useful when debugging programs.

SBASIC: for SuperBASIC fans, SBASIC is now the thing. SBASIC is a compatible replacement for SuperBASIC which runs at what most people would view as compiled speeds. This is possible due to its being written from scratch as a threaded interpreter. SBASIC adds several nice touches such as removing the need to use names with many loop control statements, nested IF statements in a single line, and integer (faster) loop controls.

Probably the best part (other than the speed) is that you can run any number of SBASIC programs fully multi-tasking with each other. Careful work has gone into how to handle the three standard windows (I will leave the details for your nightly reading of the manuals).

Misc Functionality: A few other items have been added. Since the QL community is literally world wide, you can now change the keyboard and language to suit your needs, all without having to reboot SMSQ. And if you have a favorite EPROM cartridge, you can transfer it from the original cartridge on a QL to QXL memory as a 'virtual' cartridge.

Finally, two SBASIC interface things are included. One is the FileInfo thing (Wolfgang Lenerz) which recognises and executes files ending with _sav or _bas (commonly used in the QPAC2 File Manager). The other is the "SBAS/QD" thing which is used from Jochen Merz's QD5 (or newer) editor to execute SBASIC programs from within the editor itself.

SMSQ/E - the SMSQ Enhanced Version: Now that you have a very brief overview of what SMSQ itself adds to the QDOS scene, it is time to look at the future, which by the way is NOW! SMSQ/E is one of those software projects that, while being a worthy addition to a QXL now, has even more to offer as work progresses. Be sure to read about purchasing and upgrade terms later. But first, it is time to examine what it offers to the QXL user now.

First, SMSQ/E consolidates many different pieces of software into a single unit. Many of the drivers that used to be loaded as separate software are now included (see prior SMSQ discussion). A slightly improved pointer environment is also built-in. You no longer have to load HOT_REXT, PTR_GEN, or WMAN.

SMSQ/E comes with built-in dynamic buffering for both the parallel and serial ports (it really blew me away when I first printed a twelve page basic listing with another basic formatting program in a second SBASIC window - it only took a few seconds to finish, and then SMSQ/E controlled the printout from the background with no noticeable slow down or effect on the system what-so-ever!). No more loading RAMPRT type devices drivers. And of course, you

SMSQ/E on the QXL - (CONT'D)

have control over the different buffers (dynamic or set sizing, abort and clear). Serial port control has also been expanded and includes the ability to control things like flow control (hardware, XON/XOFF, none) and buffer overflow points while the serial port is open and in use.

The next new feature is one that I also used right away, screen resizing without rebooting. As mentioned, most QXL users use higher resolution screens to pack more information in front of them at a single time. However, to change screen sizes, you had to quit everything and reboot off a different version of SMSQ that had been configured for the different screen resolution. NOT ANY MORE!

By typing in the command DISP_SIZE xxx,xxx, you can now automatically change the resolution on the screen. The first time I did it was because my eyes were getting tired (none of us work on our computers into the wee hours of the morning, do we?). So I simply stepped it down from 800x600 pixels (SVGA) to 640x480 pixels (VGA). Amazing, I could see again!

And then last night I decided to try a compatibility test. I loaded a Random-Dot Stereogram SuperBasic program (where if you cross your eyes, the image on the screen becomes three dimensional), and tried it. I'll have you know that even my eyes can not make it work when the Basic window only filled 1/4 of my screen. So I promptly typed in DISP_SIZE 512,256 (the original QL resolution), and sat cross-eyed staring at a full screen three dimensional sphere. (Notice that I typed the DISP_SIZE command in the original SBASIC windows while the Stereogram program was running in a separate set of SBASIC windows - neat). After my brain finally told my eyes to uncross, I reset my screen size to a higher resolution and went back to my not-so-fun work.

HISTORY has been added as a new virtual device. It is a single ended version of the PIPE device. It can be used to store strings and uses the concept of last in/first out. A HISTORY device can be shared between programs or kept private to a single program. Strings within a HISTORY can also be accessed by location number. I have a few new projects that can use this, but will let you come up with your own ideas! There have been a few other minor additions. One is the ability to use the INSTR command as either case dependant or independent.

Compatibility: Obviously, even though SMSQ/E does not add a large quantity of new features in its present form, the features added are useful and powerful. But what good is a powerful operating system if it is not compatible with the ten years of software you have sitting around.

Here I need to make a comment. I have some early software that hasn't been able to run since I first added anything to my original QL. There will always be a piece of software that did not follow the rules. You probably have one of those 'bad' programs in your software library. The rule of thumb, if it didn't run before, don't expect it to now!

On the other hand, all the programs that I have tried to date have run flawlessly. This includes a group of SuperBASIC programs, Text87, LineDesign, C68, etc. I was impressed when I first loaded SMSQ/E. Everything came up fine and life was good. I sure wish that I could say that about everything else I work with.

To be honest, there are a few 'quirks', but I will save them for later in this review. For now let us say that I believe that all but one of them are due to outdated software versions (mentioned in the documentation). All the quirks are minor and easy to avoid. Again, more later.

For now, let us just say that as far as compatibility goes, my congratulations go to Tony and those who helped him test everything. The amount of effort to do this must be tremendous and should be appreciated!

And the Future Holds: SMSQ/E is not finished. This is not my opinion but a statement from the manual. Even though it stands alone now as a complete and robust operating system, Tony and friends have great plans.

One of the few things that the QL world has lacked has been the ability to update partially or totally hidden windows. When a window is even partially covered, any software trying to write to it will stop running until that window is uncovered (at least in the pointer environment). This is natural considering the original QL's memory and performance limits. However, the QXL and other accelerated systems do not suffer these hardware limits.

SMSQ/E on the QXL - (CONT'D)

The first enhancement for SMSQ/E is scheduled to address this by adding background window drawing.

The next item up for improvement is to allow disk accesses to occur in the background without virtually locking up the rest of the operating system, as currently happens. The 'final' scheduled improvement (SBASIC programmers will be happy) is for the addition of a SBASIC development environment, including full debugging capabilities such as single stepping and break points.

More Observation:

Documentation: For the most part the manual is complete and fairly well organized (similar to the TKII manuals). As with any operating system documentation, the user will have to take the time to properly read it if they want to take advantage of all the new capabilities.

The manual itself is made up of a generic section covering all the SMSQ/E versions. Small sections are included for each of the different hardware platforms supported specifying the variations in the commands that are hardware specific.

Two areas that I appreciated are the sections on getting compiled SBASIC code to run and the discussion on SBASIC windows when running multiple SBASIC sessions. This could have been a tricky area for most users and obviously took a lot of thought to make really useful. The documentation may seem a bit intense, but it is well written and does explain the issues satisfactorily.

One last documentation note, there is a short but useful Trouble Shooting guide. In addition to this there are a few text files on the operating system disk. Be sure to read these for last minute updates. This is where one of my 'Quirks' was resolved.

Limitations and Problems: The only limitation that I noticed has to do with formatting floppy disks. To format a QDOS floppy disk, you must have pre-formatted it as a DOS floppy (under PC DOS). This however is very minor (most disks now come pre-formatted for DOS anyway) and after reading Tony's article in the last IQLR issue, it is understandable.

I mentioned that I ran into a few quirks with SMSQ/E. For the most part they were caused by older versions of other software that did 'No-No' things and have been fixed. The only two programs that did this to me were QPTR and PROFORMA(?). I found the QPTR version/upgrade information on one of the disk files which were previously discussed. My copy of PROFORMA is also at least one release outdated, so I suspect that its upgrade will take care of things also (sorry but could not confirm this before the article deadline).

The resulting quirk is the same with both programs. If I run either PROFORMA or QPTR (old versions), then both the DISP_TYPE and DISP_SIZE commands tell me 'not_found' for an error message. Until I receive my upgrades, I simply do any screen size work before I use either program.

The other problem seems to be one of those annoying things that can happen, no matter how careful you are in testing. On my system (not confirmed elsewhere) with V2.47 of SMSQ/E, if I set the mouse onto SER1 (properly configured), the system does not recognize SER2 as a separate device (this does work okay on SMSQ v2.47). Luckily, with the mouse in SER2, the system does recognize SER1 properly and my modem works like a charm. As most users probably have their systems configured this way anyway, they probably won't run into it. Chances are that by the time this is read, it may be fixed.

The few 'problems' found were minor. The mouse/serial port issue is workable, and the other ones are simple short term inconveniences (once this review is done, upgrades will be on order!). With the documentation even telling me where one of the problems came from, I must express my appreciation for the quality of work here.

The Bottom Line...

Just two things left to discuss, the cost/support issue and my final recommendations.

SMSQ/E on the QXL - (CONT'D)

First, you can only get SMSQ/E from Jochen Merz Software (understandable from a support and upgrade point of view). My experience with Jochen Merz has always been good. He has been reliable in not only the quality and delivery of products, but also in the support.

Since SMSQ/E is an ongoing project, its pricing structure is unique. You can buy SMSQ/E as it is now for one price and then purchase the major upgrades (at upgrade pricing) as they are available. Or you can wait and buy the 'completed' package for the same price as you would pay for the current version plus each of the scheduled upgrades. This allows you to enjoy the benefits of SMSQ/E as it stands today and get the extra features as they become available without any cost penalty.

Additionally, as many QL users work with multiple hardware platforms (I have the QXL plus a QL with the Gold Card), Merz has special pricing if you want to purchase SMSQ/E for more than one platform. And, since any product may have minor revisions (Tony seems to be constantly adding in things in response to different users!), you will be able to download or send for minor revisions at the price of a phone call or the cost of shipping and handling.

I am impressed with this software. Being a QXL owner for over a year, I have been spoiled with the performance and enhanced capabilities of SMSQ. After using SMSQ/E for the last few weeks, I must make a recommendation to upgrade further to SMSQ/E and am anticipating impatiently the upcoming version releases. A short list of the reasons are:

1. new features now (buffering, 'live' video control)
2. future capabilities (background screen/disk access)
3. compatibility and reliability
4. support
5. upgrade/new version policy

Following my own advice, now that I have SMSQ/E running on my QXL I will be ordering it for my Gold Card QL.

QXL in COMMAND

Bedford, Massachesetts - Plyesville, Maryland , USA
Al Boehm *Tom Robbins*

Newest Update

Early in November, the latest SMSQ version 2.31 arrived. It corrected the earlier glitches and both Al and Tom believe that the QXL is now fully functional. A word of thanks is due to all those at Miracle Systems and any who helped with the testing. And now in January 2.47 has arrived!! It gets better and better.

Of course, we always hope for more, and there are a lot of new keywords that we haven't fully checked out. The four parameter CURSOR error has been fixed and system crashes for either of us are rare. In fact, Tom has more crashes when multitasking in Windows then on the QXL - or the QL for that matter. Only Microsoft doesn't call them crashes - they call them a "General Protection Fault".

Configuration: The QXL now comes with the QJUMP CONFIG program that configures the SMSQ file for various screen sizes and language keyboard defaults. Note that Xchange 3.90I and TaskMan available from IQLR also make use of the higher resolution screens.

A handy way to start up the various screen sizes is to configure the files and then use PC batch files to start which one you want. For example, for QL screens and EGA screens write two batch file called QXL.BAT and QXLEGA.BAT which consist of:

IN QXL.BAT IN QXLEGA.BAT

QXL in COMMAND - (CONT'D)

```
DELETE C:\QXL\SMSQ
COPY C:\SMSQQL C:\SMSQ
QXL.EXE
```

```
DELETE C:\QXL\SMSQ
COPY C:\SMSQEGA C:SMSQ
QXL.EXE
```

QXL 5.25 disks: As mentioned before, the PC is not set up to use 720K 5.25 inch disks. Thus the QXL can not read the standard 5.25 QL disks. However, Tom has come across a MSDOS public domain utility that solves this problem and does even more. The utility consists of two programs and associated files.

MSDOS 'standard' floppy drive support is for 720 Kb/1.44 Mb 3 1/2" drives and 360K/1.2 Mb 5 1/4 inch drives. For the QXL user, this means that even if your MSDOS PC is equipped with a 5 1/4 inch 1.2 Mb drive, you can not use it. If you have a large number of files on QL 5 1/4 disks, they are not accessable to you on the QXL.

A program wriiten for the PC called FDFORMAT expands the range of disk formats a PC can use. I surmise this was written when 5 1/4 high density disks (1.2 M) were much more expensive then the 360 Kb double density disks. This program allows you, using the PC's 1.2 Mb disk drive, to format double density 5 1/4 inch disks to 720 Kb as the QL does. More signifacantly for the hardy band of QXL useres, it also allows you to format 1.2 M high density 5 1/4 inch disks to 1.44 Mb. Oddly enough, this program was written way before the QXL saw the light of day and I doubt that Tony Tebby ever heard of this program - but they work together perfectly.

For the QXL user, this means that the 5 1/4 inch 1.2 M high density PC drive can be used with the QXL and QDOS formatted disks. Using this program on your PC before ruinning QXL.EXE will allow you to read, write and format 720 K disks compatable with the QL. It will also allow you to format 1.2 M high density 5 1/4 inch disls to 1.44 M under MSDOS, which can then be cross formatted to 1.44Mb by the QXL's SMSQ operating system. Of course, these disks can not be read by a standard QL.

The FDFORMAT program was written by Christoph Hochstetter of Germany. The Menu program for FDFORMAT was written by Real LaFontaine of Canada. Both are public domain.

*** ***NOTE:*** THE 720K and 1.44M FORMATS ARE ONLY AVAILABLE IF YOU HAVE A 1.2 MB FLOPPY ON YOUR PC FDREAD - Is put in your autoexec.bat file or run from the DOS command line. It will let you read and write to 720K 5.25 inch QL disks in your 1.2M PC drive. The drive will still be fine for the 1.2M PC disks, but it will also recognize the 720K disks. The program is a small TSR.

FDFORMAT - This is a replacement for the PC FORMAT command for use formatting 720K 5.25 inch disks in your 1.2M PC drive. Don't use 1.2M HD disks - use the same double density type disks as you would use in your QL. The command line syntax is: FDFORMAT b:/f:720

An additional feature is that it lets you format 1.2M high density PC disks as 1.44 M. Thus, the 5.25 floppy can hold as much as the 3.5 floppies - very handy when you are copying disks. There is also a fancy front end for FDFORMAT called FDM.COM, but it really isn't necessary. FDREAD and FDFORMAT are being submitted to the IQLR public domain library.

Many Colors: The QXL stays with QL four color MODE 4 and eight color MODE 8. However, the QL has always had a form of dithering - stipples - that allow a two color pattern. On the QL these patterns were always noticeable and gave a chunky appearance. This is also true on the QXL in the QL mode, but in the SVGA mode these patterns are very hard to detect. In particular, stipple 3 - a cross hatch - gives very good results for all color combinations. This, in effect, gives 24 additional colors. Use this simple program in SVGA to see for yourself.

```
MODE 8
FOR i=0 to 7:FOR j=0 to 7
PAPER i,j,3: CLS:INPUT a$:REMark hit any key to go to next color.
NEXT:NEXT
```

Stipple 0 - one of one color, three of another - also gives good results for some color combinations (try MODE 8:

QXL in COMMAND - (CONT'D)

PAPER 3,2,0:CLS). The other stipples, 1 and 2, still are noticeable and give a grided effect. Stippled INK does not do so well. Letters and lines are better left in a solid color.

More SVGA: When I first got my QXL, I tended to use the EGA mode the most except for those programs that run only on the QL mode. In the EGA mode the letters are the same size as in the QL mode which depending on your monitor, is about the same size as the original QL letters. Except, of course, you have more lines (35) and characters in a line (106). With SVGA you get 130 characters and 60 lines which is really nice if you want to see how a full page will look on a wide body printer. However, these characters are small. With my eyesight being what it is, I am amazed I can read them at all. I think it is the high quality (Vision 15) monitor that allows this clarity.

However after several hours of viewing and programming, the tiny letters are too tiny for me. So I use CSIZE 1,1. This gives 54 characters and 30 lines and a very easy to read screen. Why not just stick with the EGA? The answer is in the graphics. The SVGA have to be seen to be appreciated. Furthermore, these graphics are easily dumped to my printer!

Screen Dumps: The Gold Card, Super Gold Card, and the QXL have built in screen dumps for a large variety of printers. Since many of you don't have a manual, a condensed list may be of use. To set up the dump use:

Generic	Example
SDP_DEV 'Output Device'	SDP_DEV 'PAR'
SDP_SET Printer,scale,inverse,random	SDP_SET 1,1,1,0

Where Output Device can be PAR, SER1, or a file - FLP1_Picture. Scale is 1,2,or 3 which varies the size of the printer picture. Inverse=0 means black is black on the printer which can use a lot of ink, so recommended is inverse=1 which makes black on the monitor print as white. Random=1 adds random dots to give grey scales for various colors on black and white printers. Works pretty good. The printers are specified by their number:

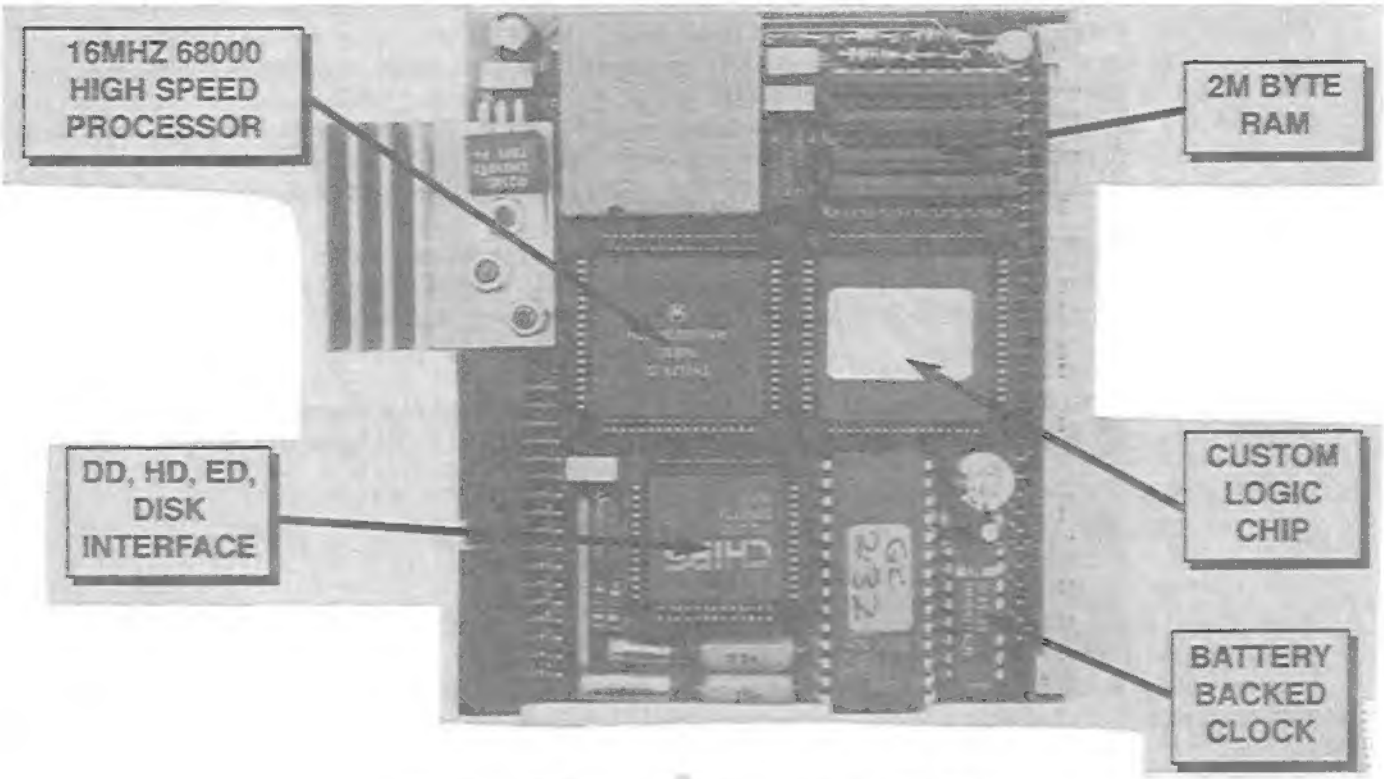
Brother HR4	9	Epson MX 100	19
Brother 8056	18	Fastext 80	22
Cannon PJ1080A	14	MT-80	23
Centronics 739	15	OKI Microline 82/84	21
C.Itoh 7500	16	OK Writer	21
Epson FX 80	2	Olivetti JP101	10
Epson FX100 (wide)	3	Seikosha GP-100A	11
Epson JX 80	4	Seikosha GP-250X	12
Epson LQ2500 8pin	5	Seikosha GP-700A	13
24pin	6	Star SJ-144	8
8pin color	7	Tandy DMO 105	20
24pin color	8	Toshiba TH2100H 24pin	17

If your printer isn't here, don't despair. Mine wasn't either. But I looked for a printer it emulated. Still not there. But some printers that had similar numbers were. I tried them all and found out the my Star SJ-144 using 8 has some very pretty dumps especially the SVGA screens. I was also impressed by the wide carriage dumps on my ALPS PG1000G (emulates FX100) printer. By the way, when you do find out what number works with your printer that isn't listed, send it in to IQLR. We'll update the list in a future issue. I expect Miracle will now and then add some additional printers.

Note from Tom: The NEC P7 uses the Epson LQ 2500 24 pin screen dump (number 8). The printer dump is started by the command SDUMP. A hot key can also be set up with the command SDP_KEY p (or some other letter than p if you wish). Then when ALT and p are pressed, the screen is dumped. It is also possible to dump part of a screen, but I haven't tried that yet.

By the time you read this, SMSQ 2.47 will be in general circulation. Some new features are: The QXL will accept file names with the "." seperator on QL formatted disks. The QXL now handles subdirectories on MSDOS floppy disks. There is a reset key sequence for the QXL: ALT-SHIFT-CONTROL-TAB. ***And one last quick note***: The QXL runs quite nicely under OS/2.

MIRACLE SYSTEMS



QL GOLD CARD

Recycled Gold Card £100 inc. (£90 outside EU)

This is the expansion that has been revolutionising the QL. It is very easy to fit , it simply plugs into the expansion port at the left hand of the QL, and once fitted it will instantly increase the execution speed of the QL by about 4 times due to the presence of a 16MHz 68000 on board. There is 2M of fast 16 bit RAM of which QDOS sees a contiguous 1920K. The remainder is used for shadowing the QL's ROM and display memory and for the GOLD CARD's own code.

There is a disk interface which can access 3 mechanisms (4 with the DISK ADAPTER) of three different densities, DD (double density, 720K), HD (high density, 1.44M) and ED (extra high density, 3.2M) in any mix. The disk interface connector is the same type that was fitted to the Trump Card so most QL compatible disk drives can be used.

> Please note: that DD drives still give a capacity of 720K per diskette.
> Our DUAL ED DISK DRIVE allows the GOLD CARD to access DD, HD and ED diskettes.

Another feature is the battery backed clock. When the QL is switched on the contents of the clock are copied into the QL's clock so that the time and date are correct. The firmware in the ROM gives the GOLD CARD all the functionality of the Trump Card like TOOLKIT II and there is a sub-directory system for floppy and RAM disks.

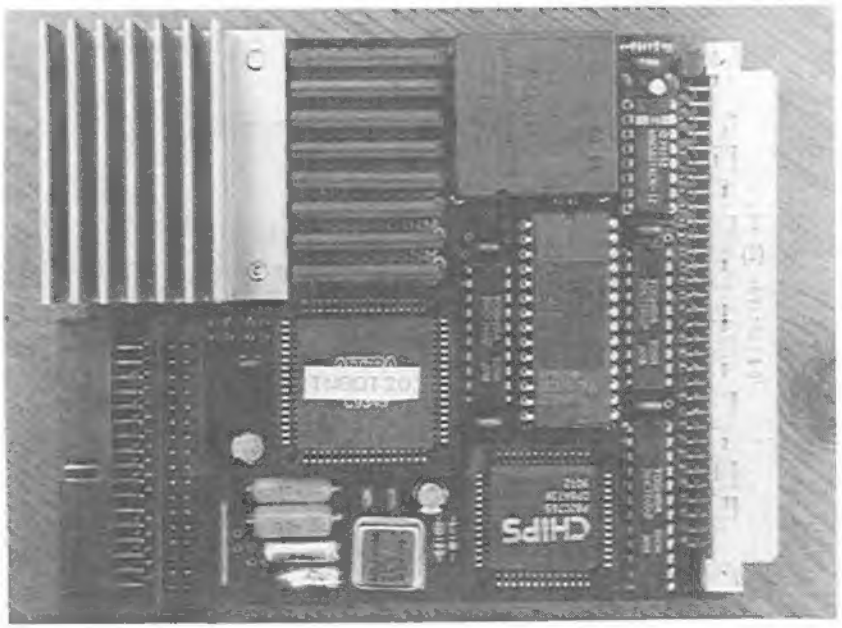
Physically the GOLD CARD is about half the size of the TRUMP CARD and so fits almost all within the QL. Its current consumption is well under allowable maximum so no special power supply is required. The GOLD CARD comes with a 14 day money back guarantee and a 1 year warranty.

MIRACLE SYSTEMS

SUPER GOLD CARD

"The Pathway to Future QL Development"

Briefly...



- *3 Times Faster*
- *68020 processor*
- *4M bytes of RAM*
- *CENTRONICS port*
- *Supports 4 Disk Drives*
- *2 Year warranty*

What is it ?

The SUPER GOLD CARD is the first major revision of our highly successful Gold Card. We have replaced the 68000 processor with the 68020 so programs run about 3 times faster and have expanded the memory to 4M bytes. Additional improvments include a fast CENTRONICS printer port, 2 double disk drive ports, virtually crash-proof clock and a socket to optionally connect 5V. We also supply a 3 meter Centronics printer cable at no additional cost.

The deal...

£275 including VAT - (£240 outside EU)

(Includes postage, a 14 day money back guarantee and a 2 year warranty.)

or you can send us your GOLD CARD and:

£175 including VAT - (£150 outside EU)

You can also deduct a further £15 for a returned QL CENTRONICS and/or £10 for a DISK ADAPTER.

We are happy to accept payment by sterling cheque made payable to "MIRACLE SYSTEMS", or by quoting your MASTERCARD/VISA/SWITCH credit card number and expiry date (SWITCH card holders please also quote issue number).

Recycled Items...

Gold Card £100 (outside EU £90) QL Centronics £ 15 Disk Adapter £ 10

(Recycled items carry a 1 year warranty.)

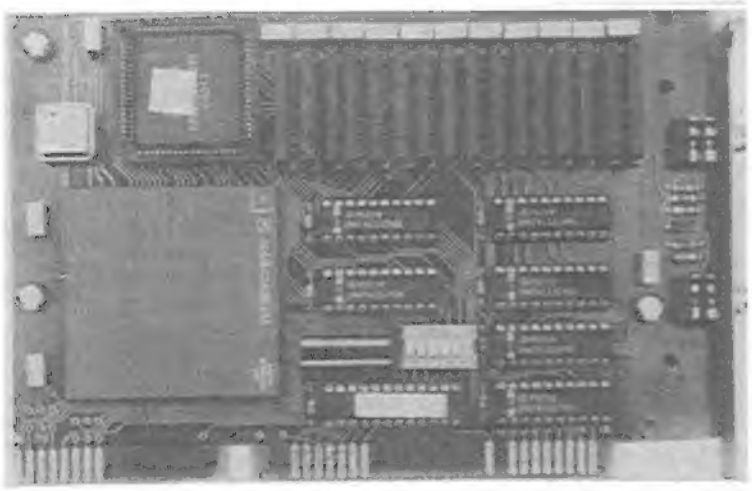
TELEPHONE/FAX: (01454) 883 602

MIRACLE SYSTEMS LTD - 20 Mow Barton, Yate, Bristol, BS17 5NF, UK

MIRACLE SYSTEMS

QXL
Now With
SBASIC

SuperBasic Compatible Interpreter



- *68EC040*
- *4M or 8M of RAM*
- *Multitasking SBASIC*
- *QL Network ports*
- *Toolkit II*
- *QDOS or MSDOS floppies*
- *Uses PC's keyboard, floppy & hard disks, parallel/serial ports and mouse.*

This is the card that plugs into a standard 8 or 16 bit ISA slot on a PC and allows the PC to run QL programs - FAST. A new QDOS compatible operating system from Tony Tebby called SMSQ, which is supplied on a disk, includes Toolkit II and gives you the familiar QL environment. SMSQ includes SBASIC a multitasking SuperBasic compatible interpreter.

Installation is simple; plug the QXL into a spare slot and copy 2 files from the supplied disk onto the hard drive and you're ready to go. From the DOS prompt type QXL and the PC will transform itself into a QL before your very eyes. If at any stage you wish to return to DOS just press CTRL-ScrollLock. You can later resume the QL session by typing QXL/ which takes you back to where you left off. For POINTER ENVIRONMENT programs SMSQ can be configured to handle 3 screen resolutions in addition to the standard 512x256 QL screen. Your PC must have EGA or VGA graphics. EGA allows 640x350 whereas VGA also allows 640x480. Most SVGA cards will allow SMSQ to use 800x600 as well.

PRICING:

QXL (4M)	£280 including VAT - (£245 outside EU)
QXL (8M)	£395 including VAT - (£345 outside EU)

or you can send us your GOLD CARD and

£180 including VAT - (£155 outside EU) for a 4M - QXL

£295 including VAT - (£255 outside EU) for a 8M - QXL

You can also deduct a further £15 for a returned QL CENTRONICS and/or £10 for a DISK ADAPTER.

We are happy to accept payment by sterling cheque made payable to "MIRACLE SYSTEMS", or by quoting your MASTERCARD/VISA/SWITCH credit card number and expiry date (SWITCH card holders please also quote issue number).

TELEPHONE/FAX: (01454) 883 602

MIRACLE SYSTEMS LTD - 20 Mow Barton, Yate, Bristol, BS17 5NF,